

Міністерство освіти і науки України
Запорізький національний університет

**С. В. ЧОПОРОВ, О. В. КУДІН, Є. В. ПАНАСЕНКО,
Д. Д. ГРИЩАК, М. С. ІГНАТЧЕНКО**

**МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ
ІНЖЕНЕРНОГО АНАЛІЗУ
ОБ’ЄКТІВ АЕРОКОСМІЧНОЇ ТЕХНІКИ НА БАЗІ
ХМАРНИХ ТЕХНОЛОГІЙ**

Монографія

За наук. ред. С. В. Чопорова

Херсон
Видавничий дім «Гельветика»
2020

УДК 519.8:004.94:629.78

МЗ4

РЕЦЕНЗЕНТИ:

Максименко-Шейко К. В., доктор технічних наук, старший науковий співробітник, учений секретар Інституту проблем машинобудування ім. А. М. Підгорного НАН України;

Субботін С. О., доктор технічних наук, професор, завідувач кафедри, завідувач кафедри програмних засобів Національного університету «Запорізька політехніка».

Рекомендовано до друку Вченою радою Запорізького національного університету, протокол № 2 від 29 вересня 2020 року.

МЗ4 Математичне забезпечення інженерного аналізу об'єктів аерокосмічної техніки на базі хмарних технологій: монографія / С. В. Чопоров, О. В. Кудін, Є. В. Панасенко, Д. Д. Грищак, М. С. Ігнатченко [за наук. ред. С. В. Чопорова]. Херсон: Видавничий дім «Гельветика», 2020. 300 с.

ISBN 978-966-992-217-5

Монографію присвячено висвітленню методів математичного моделювання поведінки об'єктів аерокосмічної техніки для розробки хмарних САПР. Розглянуто методи геометричного моделювання суцільних тіл і оболонок, генерації дискретних моделей, скінченно-елементного аналізу, адаптивного моделювання, обробки чисельних результатів, особливості програмної реалізації.

Призначено для наукових співробітників, аспірантів та інженерів, робота яких пов'язана з проектуванням та дослідженням поведінки технічних об'єктів на базі чисельних методів.

Л.: 175. Табл.: 18. Бібліогр.: 345.

Mathematical Methods in Cloud Computer-Aided Engineering for the Aerospace Objects Analysis / S. V. Choporov, O. V. Kudin, Y. V. Panasenko, D. D. Gristchak, M. S. Ignatchenko [edited by S. V. Choporov]. Kherson: Publishing House Helvetica, 2020, 300 p.

This monograph is devoted to problems of space technique objects mathematical modelling for cloud CAE development. The monograph includes methods for geometric modelling of solids and shells, two- and three-dimensional meshing, the finite element analysis, adaptive modelling, numerical results processing, and key steps in software implementation.

The edition is suggested for researchers, post-graduate students and engineers who work in the areas of computer-aided engineering and numerical methods.

Images: 175. Tables: 18. References: 345.

ISBN 978-966-992-217-5 © Чопоров С. В., Кудін О. В., Панасенко Є. В., Грищак Д. Д., Ігнатченко М. С.

© Запорізький національний університет

© Видавничий дім «Гельветика»

ЗМІСТ

Вступ.....	7
Розділ 1. Препроцесор.....	10
1.1. Постановка проблеми розробки препроцесора інженерного аналізу об'єктів аерокосмічної техніки.....	10
1.2. Аналіз сучасних САПР машинобудування та будівництва.....	12
1.3. Аналіз методів подання форм об'єктів у САПР машинобудування та будівництва.....	13
1.4. Аналіз методів побудови дискретних моделей форм об'єктів у САПР машинобудування та архітектури.....	15
1.4.1 Структуровані дискретні моделі форм об'єктів.....	16
1.4.2 Неструктуровані дискретні моделі форм об'єктів.....	24
1.4.3 Дискретні моделі оболонкових конструкцій.....	30
1.5. Аналіз методів оптимізації дискретних моделей форм об'єктів.....	33
1.5.1 Методи оптимізації внутрішніх вузлів та елементів.....	33
1.5.2 Методи оптимізації межових вузлів та елементів.....	34
1.6. Подання форм об'єктів у хмарних САПР з використанням неявних функцій.....	36
1.6.1 Моделювання двовимірних форм об'єктів на базі неявних функцій. .42	
1.6.1.1 Примітиви.....	42
1.6.1.2 Приклади функціонального подання креслень двовимірних форм об'єктів.....	47
1.6.2 Моделювання тривимірних форм об'єктів на базі неявних функцій. .52	
1.6.2.1 Примітиви.....	52
1.6.2.2 Приклади функціонального подання креслень тривимірних форм об'єктів.....	55
1.7. Гібридний підхід до подання форм об'єктів машинобудування.....	63
1.7.1 Двовимірні форми.....	65
1.7.2 Тривимірні форми.....	71
1.8. Подання оболонкових об'єктів і конструкцій.....	82
1.9. Подання дискретних моделей у САПР.....	84
1.9.1 Формальне визначення дискретної моделі форми об'єкта машинобудування.....	86
1.9.2 Операції вставки та видалення, пошуку сусідів.....	87
1.9.3 Структури даних.....	91

1.10. Дискретні моделі меж об'єктів машинобудування.....	94
1.10.1 Проекційний метод генерації дискретних моделей меж об'єктів, форма яких задана функціонально.....	97
1.10.1.1 Загальний алгоритм методу.....	97
1.10.1.2 Алгоритм пошуку проекцій вузлів на межу об'єкта.....	99
1.10.1.3 Приклади застосування.....	100
1.10.1.4 Двовимірний випадок.....	104
1.10.1.5 Аналіз результатів методу.....	105
1.10.2 Метод згладжування координат вузлів дискретних моделей меж об'єктів, форма яких задана функціонально.....	106
1.10.2.1 Функціонал відстані-довжини.....	107
1.10.2.2 Локальний алгоритм пошуку мінімуму функціоналу відстані-довжини.....	111
1.10.2.3 Глобальний алгоритм пошуку мінімуму функціоналу відстані-довжини.....	112
1.10.2.4 Приклади застосування методу.....	113
1.10.2.5 Двовимірний випадок.....	117
1.10.2.6 Аналіз результатів методу.....	118
1.10.3 Метод локальних перебудов елементів дискретних моделей меж об'єктів.....	119
1.11. Дискретні моделі форм об'єктів на базі солідів.....	124
1.11.1 Дво- і тривимірна тріангуляція.....	124
1.11.1.1 Алгоритм методу тріангуляції.....	125
1.11.1.2 Алгоритм оптимізації чисельних характеристик дискретної моделі.....	133
1.11.2 Метод генерації дискретних моделей на базі елементів типу солід для форм об'єктів, заданих функціонально.....	139
1.11.3 Згладжування Лапласа для внутрішніх вузлів.....	147
1.11.4 Метод згладжування координат внутрішніх вузлів дискретних моделей форм об'єктів.....	152
1.11.5 Особливості використання методу для побудови дискретних моделей поверхонь об'єктів, поданих гібридно.....	161
Розділ 2. Процесор.....	164
2.1. Дослідження впливу форми скінченного елемента на точність розв'язку.....	164
2.1.1 Двовимірні задачі дослідження напружено-деформованого стану...165	
2.1.1.1 Згин консолі, навантаженої на кінці.....	166
2.1.1.2 Згин балки рівномірним навантаженням.....	168
2.1.2 Тривимірні задачі дослідження напружено-деформованого стану...170	

2.2. Скінченно-елементне моделювання оболонок ракетної техніки.....	173
2.2.1 Моделювання тонких і середніх оболонок як сукупності плоских елементів із ізотропного матеріалу.....	174
2.2.1.1 Дослідження прогину циліндричної оболонки.....	179
2.2.1.2 Дослідження напружено-деформованого стану баку.....	181
2.2.2 Оболонки зі змінною товщиною.....	182
2.2.3 Температурні деформації в оболонках.....	183
2.2.4 Пружно-пластичний випадок на базі методу змінної пружності.....	185
2.3. Скінченно-елементне моделювання багатошарових оболонок ракетної техніки із транстропних матеріалів.....	187
2.4. Приклад дослідження напружено-деформованого стану тришарової оболонки ракети-носія, яка знаходиться під дією комбінованого навантаження.....	189
2.5. Аналітико-чисельні методи у хмарних САПР.....	193
2.5.1 Наближене аналітичне розв'язання задачі стійкості тришарової конічної оболонки при комбінованому навантаженні.....	193
2.5.1.1 Постановка задачі.....	194
2.5.1.2 Фізичні характеристики тришарової оболонки.....	195
2.5.1.3 Розв'язувальне диференціальне рівняння.....	197
2.5.1.4 Застосування ВКБ методу для розв'язання рівняння.....	201
2.5.1.5 Застосування гібридного ВКБ-Гальоркін методу для розв'язання рівняння.....	202
2.5.1.6 Застосування методу скінченних різниць для розв'язання рівняння.....	204
2.5.1.7 Порівняння результатів.....	205
2.5.1.8 Гранична поверхня, що відокремлює область стійкості від області нестійкості.....	206
2.5.2 Вплив параметрів конічної тришарової оболонки на стійкість.....	207
2.5.3 Деформування циліндричних тришарових оболонок при дії зовнішнього тиску.....	211
Розділ 3. Постпроцесор.....	215
3.1. Метод генерації адаптивних дискретних моделей форм об'єктів, заданих функціонально.....	215
3.2. Метод формування множини елементів для розбиття.....	223
3.3. Візуалізація чисельних даних на поверхні конструкції.....	232
Розділ 4. Каркас програмного забезпечення.....	236
4.1. Шаблони проектування для об'єктно-орієнтованих САПР машинобудування.....	236
4.1.1 Декомпозиція та ідентифікація.....	237

4.1.2 Каталог шаблонів проектування.....	238
4.1.2.1 Шаблон проектування «UI–Model–Analysis».....	238
4.1.2.2 Шаблон проектування «Representation–Mesh».....	239
4.1.2.3 Шаблон проектування «Element–Node».....	241
4.1.2.4 Шаблон проектування «FEA Problem».....	242
4.2. Методи генерації та обробки дискретних моделей форм об'єктів на базі технологій паралельних обчислень.....	244
4.2.1 Метод генерації дискретних моделей форм об'єктів з використанням паралельних обчислень.....	244
4.2.2 Скінченно-елементний аналіз з використанням технологій паралельних обчислень.....	246
4.3. Проблемно-орієнтована мова моделювання форм об'єктів.....	253
4.3.1 Базовий синтаксис.....	253
4.3.2 Допоміжні функції й операції.....	256
4.3.3 Основні геометричні примітиви.....	257
4.3.4 Генерація дискретних моделей.....	259
4.3.5 Опис скінченно-елементних задач.....	262
4.4. Об'єктно-орієнтована модель САПР для паралельних комп'ютерних систем.....	267
4.5. Модель роботи методів у хмарному середовищі.....	268
4.5.1 Лінійна модель менеджера черг.....	269
4.5.2 Модель менеджера черг у розподіленій комп'ютерній системі.....	270
4.5.3 Особливості програмної реалізації.....	271
Післямова.....	275
Перелік посилань.....	278

ВСТУП

Актуальні тенденції у розвитку таких галузей як будівництво, авіа-, машино-, судно- і ракетобудування направлені на підвищення якості об'єктів, що випускаються, а також зменшення їх собівартості. Для цього використовують методи проектування, включно експериментальне та математичне моделювання, конструювання тощо.

Важливою частиною процесу проектування складних технічних об'єктів є дослідження їх експлуатаційних характеристик. Для цього використовують моделі різних видів: графічні, математичні, натурні тощо. Застосування математичного моделювання дозволяє зменшити витрати на натурні дослідження об'єктів, а також зробити прогнози відносно режимів експлуатації.

Часто на практиці об'єкти машинобудування мають складну форму. Об'єкти бувають як суцільними тілами, так і тонкостінними оболонками із композиційних матеріалів. Їх натурні випробування потребують підготовки умов, що відповідають режимам експлуатації об'єкта, і, як наслідок, є вельми вартісними. У таких випадках для підвищення ефективності процесу проектування об'єктів машинобудування використовують математичне моделювання на базі комп'ютерної техніки. Факторами, що впливають на ефективність процесу проектування об'єктів машинобудування, стають точність математичних моделей та час, необхідний на комп'ютерне моделювання.

Програмне забезпечення, яке забезпечує комп'ютерне моделювання об'єктів, що проектують, називають системами автоматизованого проектування і розрахунку (або скорочено САПР або CAD¹). У межах САПР програми, які автоматизують інженерні розрахунки та моделювання об'єктів проектування, називають системами автоматизації інженерних розрахунків (скорочено CAE²).

Математичні моделі об'єктів, отримані на стадії науково-дослідних робіт проектування у машинобудуванні та будівництві, часто базуються на складних диференціальних та/або інтегральних рівняннях, для розв'язку яких потрібно використовувати чисельні методи. В основу багатьох поширених чисельних методів (наприклад, методів граничних елементів, скінченних елементів, скінченних різниць тощо) покладено ідею переходу від неперервної моделі до її дискретного аналогу. В одному з найбільш поширених чисельних методів, методі скінченних елементів (як і у деяких інших), для визначення стану

1 Англійською Computer-Aided Design або CAD.

2 Англійською Computer-Aided Engineering або CAE

об'єкта його форма подається простими фігурами (наприклад, трикутниками або тетраедрами), для котрих обчислюється апроксимація розв'язку вихідного рівняння. Відповідно, важливим кроком такого аналізу є побудова дискретної моделі форми об'єкта, яка дозволить отримати розв'язок із потрібною точністю.

Отже, у структурі сучасних САЕ машинобудування й архітектури умовно можна виділити три основні частини: препроцесор, процесор і постпроцесор.

Під препроцесором, зазвичай, розуміють засоби комп'ютерного опису моделей об'єктів проектування. Математична модель або креслення об'єкту є вхідним параметром препроцесора, який генерує дискретну модель на другому кроці. Отримана дискретна модель є вхідним параметром для процесора чисельного методу.

Процесор – ядро САПР – це засоби комп'ютерного аналізу моделей об'єктів проектування. Основою комп'ютерного аналізу в САПР машинобудування й архітектури є чисельні методи, найбільш поширеними серед яких є методи скінченних елементів, скінченних різниць, граничних елементів, скінченних об'ємів і безсіткові методи³. Чисельний аналіз поведінки сучасних конструкцій на базі цих методів зумовлює необхідність розв'язання розріджених систем лінійних алгебраїчних рівнянь великого розміру⁴. У результаті реалізація сучасних САЕ пов'язана з необхідністю використання технологій паралельних і розподілених обчислень.

Результатом роботи процесора САПР машинобудування й архітектури є велика кількість значень шуканої величини (наприклад, переміщень) у вузлах дискретної моделі. Обробку отриманих значень покладено на постпроцесор, який готує данні до візуалізації результатів у доступній для інженера-дослідника формі, а також забезпечує оптимізацію моделей препроцесора і процесора.

Стрімкий розвиток мережевих технологій зумовив перетворення програмного забезпечення з продукту на послугу, роботу якої зазвичай забезпечує певна кількість фізичних або віртуальних комп'ютерів (хмара). Комп'ютери у хмарі взаємодіють у процесі розподілу навантаження при вирішенні задач користувача. Водночас користувач «бачить» лише одну точку – веб або мобільний застосунок, що взаємодіє з хмарним сервісом.

Оскільки використання хмарних технологій передбачає наявність загальнодоступної мережі серверів, на які виконують обчислення та надсилають результати обчислень на комп'ютер користувача, хмарні САПР зі звичайного програмного продукту перетворюється на вид інформаційної послуги. Типовий процес роботи хмарної САПР буде складатися з таких кроків.

3 Англійською meshfree methods.

4 Системи лінійних алгебраїчних рівнянь, з кількістю невідомих, що більша за мільйон, з переважною більшістю нулів у матриці.

1. Інженер-проектувальник формує модель об'єкта та передає її через мережу у хмару.

2. Доступний (найменш завантажений) препроцесор приймає модель, виконує лінгвістичний аналіз і верифікацію параметрів, якщо модель та всі параметри є валідними, то починає роботу одне з ядр процесора.

3. По закінченню роботи процесора результати його обчислень обробляє постпроцесор та необхідний інженеру зріз даних хмара відправляє через мережу до замовника.

4. На стороні клієнта спеціальні засоби виконують візуалізацію даних.

Запропонована вашій увазі монографія містить основні теоретичні концепції, теорій, методи та підходи, пов'язані з реалізацією хмарної САПР. У першому розділі розглянуті основні методи побудови дискретних моделей для препроцесора. Другий розділ присвячено реалізації процесору САПР на базі методу скінченних елементів. Третій розділ містить основні ідеї щодо постпроцесора. У четвертому розділі розглянуто деякі особливості програмної реалізації.

РОЗДІЛ 1. ПРЕПРОЦЕСОР

1.1. Постановка проблеми розробки препроцесора інженерного аналізу об'єктів аерокосмічної техніки

Важливою частиною процесу проектування складних технічних об'єктів є дослідження їх експлуатаційних характеристик. Для цього використовують моделі різних видів: графічні, математичні, натурні тощо. Застосування математичного моделювання дозволяє зменшити витрати на натурні дослідження об'єктів, а також зробити прогнози відносно режимів експлуатації. Стан технічних об'єктів (наприклад, напружено-деформований) часто моделюється системами диференціальних рівнянь з частинними похідними, які не мають аналітичного розв'язку. У такому випадку використовують чисельні методи, які засновані на ідеї переходу від неперервної моделі об'єкта до дискретної. У результаті виникає необхідність автоматичної побудови дискретних моделей, які дозволяють розв'язати системи диференціальних рівнянь із заданою точністю.

Побудова дискретних моделей об'єктів машинобудування пов'язана з необхідністю опису їх форм. Для цього використовують різні подання форм об'єктів: граничне, конструктивна блокова геометрія, функціональне тощо. Функціональне подання, яке ґрунтується на використанні неявних функцій, є найбільш універсальним і теоретично дозволяє описати форму об'єкта довільної складності. Неявний опис форми об'єкта дозволяє класифікувати положення точки простору відносно його межі, але потребує розробки методів генерації дискретних моделей як об'єднання елементарних фігур (meshing).

При використанні дискретних моделей форм об'єктів спрощено процес дослідження об'єкта, що проектується, можна подати такою послідовністю кроків:

- 1) побудова математичної моделі форми об'єкта;
- 2) автоматична генерація еквівалентної дискретної моделі;
- 3) комп'ютерний аналіз необхідних характеристик.

Отримані у результаті комп'ютерного аналізу параметри поведінки досліджуваного об'єкта можна використовувати для прийняття проектних рішень або оптимізації характеристик проектованого об'єкта (наприклад, для мінімізації маси, оптимізації параметрів армування обшивок тощо). У

останньому випадку концептуальна схема автоматизації етапу науково-дослідних робіт може бути подана алгоритмом (рис. 1.1), у якому на першому кроці формалізується модель. Математична модель виступає як вхідний параметр препроцесора, який генерує дискретну модель на другому кроці. Отримана дискретна модель є вхідним параметром для процесора чисельного методу. У результаті чисельного аналізу на третьому кроці отримують необхідні параметри стану об'єкта, що проектується. Параметри стану є вхідними для модуля оптимізації, який коригує роботу препроцесора (наприклад, варіюючи товщиною об'єкта) поки не будуть отримані оптимальні значення параметрів стану об'єкта або не буде забезпечена потрібна точність моделювання.

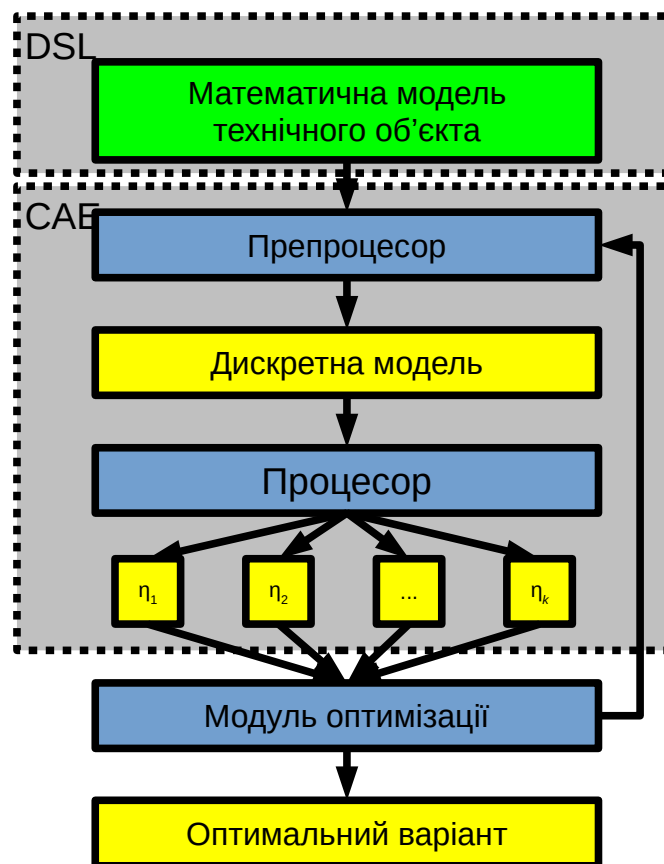


Рисунок 1.1 – Концептуальна схема автоматизації CAE машинобудування або архітектури

Водночас необхідно забезпечити введення модельної інформації проектувальником зрозумілим для нього способом. Актуальним напрямом пошуку розв'язку задачі формалізації вихідних даних в САПР є розробка лінгвістичного забезпечення – проблемно-орієнтованих мов. Вхідна мова повинна включати в себе дві складові: мову опису об'єктів і мову опису завдань. Мова опису об'єктів, у контексті розробки процесора хмарних CAE машинобудування або архітектури, повинна надавати засоби опису форм об'єктів та їх фізико-механічних властивостей. Мова опису завдань повинна

включати засоби опису певних проектних операцій або процедур (наприклад, розрахунку з використанням методу скінченних елементів, оптимізації тощо).

1.2. Аналіз сучасних САПР машинобудування та будівництва

На сьогодні розроблено значну кількість як вітчизняних так і зарубіжних САПР, які надають засоби для чисельного аналізу об'єктів машинобудування та будівництва. Популярними CAE є ANSYS, NASTRAN, PATRAN, ABAQUS, LS-DYNA, CATIA, PRO / ENGINEER, COSMOS, LIRA тощо.

Поширеною є система ANSYS [1], яка дозволяє розв'язувати значний клас задач інженерного аналізу. Серед них задачі дослідження напружено-деформованого стану при статичному і динамічному навантаженнях, механіки руйнування [2], переносу тепла [3], течії газів і рідин [4] тощо. Інструментарій дозволяє використовувати лінійні і нелінійні скінченні елементи, бібліотеку матеріалів і вибирати процесор методу скінченних елементів (МСЕ). Зокрема, як процесор МСЕ в системі ANSYS доступний LS-DYNA [5–6], який дозволяє розв'язувати задачі динамічної взаємодії твердих тіл, задачі удару, захисту лопаток, штампування та виливки.

Система MSC NASTRAN дозволяє скінченно-елементний аналіз напружено-деформованого стану, запасу міцності, власних частот і форм коливань, стійкості, усталені та несталі динамічні процеси, задач теплопередачі, акустичних явищ [7] тощо. Цю систему використовують для скінченно-елементного аналізу композитів [8], ламінатів [9], аерокосмічних матеріалів [10] тощо. Ця САПР дозволяє використовувати високопродуктивні обчислення [11].

САПР Patran [12] є інтегрувальною для систем інженерного аналізу, моделювання та проектування. Дана САПР забезпечує можливість інтегрування систем геометричного та скінченно-елементного моделювання, обробки результатів обчислень [13], дослідження міцності у лінійній [14] та нелінійній [15] постановках, задач динаміки [16] тощо.

Скінченно-елементний програмний комплекс ABAQUS можна розглядати як САПР функціонального проектування (CAE). Ця система є вельми гнучкою, вона підтримує розширення на мові програмування Python [17]. Її використовують для розв'язання задач контактної взаємодії [18], механіки руйнування [19], композитних матеріалів [8], переносу тепла [20] тощо.

До типу CAE також відноситься система COSMOS [21–22], в основу якої покладено МСЕ. Ця САПР дозволяє розв'язувати статичні та динамічні задачі

механіки твердого тіла, досліджувати процеси переносу тепла, динаміки газів і рідин тощо.

Вітчизняні програмні комплекси (ПК) ЛІРА [23–24] та МОНОМАХ [25], які розроблені під керівництвом професора А. С. Городецького, є відомими САПР будівництва та машинобудування. ПК ЛІРА призначений для проектування та розрахунку (у тому числі дослідження міцності та стійкості) конструкцій будівництва та машинобудування. ПК МОНОМАХ – для автоматизованого проектування багатоповерхових і каркасних залізобетонних конструкцій. В основу підсистем інженерного аналізу обох ПК покладено МСЕ (метод переміщень).

Інший вітчизняний ПК МІРЕЛІА+ [26–30] розроблений під керівництвом професора Киричевського В. В. Цей ПК написано на мові FORTRAN. Він дозволяє автоматизувати скінченно-елементний аналіз конструкцій з еластомерних матеріалів.

САПР FORTU-FEM [31–37] базується на використанні об'єктно-орієнтованої мови FROTU-3 для формалізації скінченно-елементних моделей технічних об'єктів і процесів. Водночас користувач може самостійно визначити функціонал, машинізація якого буде відповідати стану системи, що моделюється.

Комп'ютерна система «POLYE-RL» [38–39] призначена для розв'язку крайових задач математичної фізики з використанням теорії R-функцій, у тому рахунку на об'єктах фрактальної геометрії.

Потрібно зазначити, що розроблено також інші САПР, засновані на МСЕ. Серед них можна виокремити, наприклад, ADINA, HyperWorks, COMSOL Multiphysics, а також системи з відкритим вихідним кодом: MFree2D, FreeFem++, Elmer тощо [40–42].

Популярність МСЕ у системах САЕ доводить актуальність вирішення проблеми підвищення ефективності проектування об'єктів машинобудування проблеми шляхом збільшення точності скінченно-елементного аналізу.

1.3. Аналіз методів подання форм об'єктів у САПР машинобудування та будівництва

Реалізація підходів і технік генерації дискретних моделей форм об'єктів залежить від схеми подання останніх. Під схемою подання множини форм об'єктів $\mathbf{O}$ множиною $\mathbf{L}$ розуміють відношення r , яке встановлює відповідність між $\mathbf{O}$ і $\mathbf{L}$ [43]. Якщо $\mathbf{O}$ – множина всіх можливих форм об'єктів, а $\mathbf{L}$ – множина

формальних описів моделей геометричних об'єктів, то схема подання r встановлює відношення між формальним описом і формою об'єкта. Схему подання можна застосовувати у САПР машинобудування, якщо відношення r , що реалізує схему подання, є бієкцією.

У дослідженнях з комп'ютерного моделювання форм об'єктів, виконаних в роботах А. G. Requicha [45–48], виділені вісім основних схем подання.

1. Інженерні креслення – вельми поширений інструмент «спілкування» між інженерами шляхом подання об'єкта у вигляді набору двовимірних проекцій. Основним їх недоліком є можливість неоднозначного трактування деяких елементів, що призводить до складності їх формалізації.

2. Каркасне подання [49–50] використовувалося в ранніх САПР машинобудування для візуалізації об'єктів їх ребрами (каркасом). Воно допускає неоднозначність інтерпретації, отже, його застосування обмежене.

3. Подання екземплярами примітивів засноване на формуванні колекції параметризованих моделей форм об'єктів. Кожен примітив власною назвою та списком параметрів подає сімейство форм. Основний недолік такого підходу – неможливість формування нових примітивів з існуючих.

4. Воксельне подання базується на перерахуванні кубічних комірок, які повністю належать об'єкту. Це подання застосовується архітектурі, де об'єкти часто мають блокову структуру [51]. Проте, загалом, воно подає форму з деякою точністю, яка збільшується зі зменшенням розміру комірки, але водночас зростає вимога до обчислювальних ресурсів.

5. Дискретні моделі подають форми об'єктів множинами простих фігур, які не перетинаються і можуть мати спільні вершини, ребра та грані. Поверхня об'єкта апроксимується гранями елементів. Основні області застосування цього подання – візуалізація і чисельний аналіз стану об'єктів.

6. Конструктивна суцільна (блокова) геометрія [44–48, 52–55] (Constructive Solid Geometry, CSG) об'єднує сімейство схем подання твердих тіл як результат булевих операцій для комбінування примітивів або суцільних блоків. Семантично модель тіла розглядається як дерево теоретико-множинних операцій (об'єднання, перетин, різниця) і примітивів, поданих суцільними підмножинами евклідового простору. Примітивами, зазвичай, є найпростіші геометричні фігури (куб, циліндр, призма, піраміда, сфера, конус), що ускладнює їх застосування для моделювання нестандартних тіл.

7. Схеми подання розгорткою засновані на простому спостереженні, що рух плоского множини через простір може утворити «об'єм», що подається рухомим об'єктом і траєкторією його руху. Найчастіше так отримують тіла обертання або призми з різними перетинами. Такі техніки вельми наочні, однак їх застосування обмежене.

8. Граничне подання (Brep) [56–60] розглядає форму об'єкта як композицію його меж. Межі подають прямолінійними (наприклад, трикутниками) або криволінійними ділянками (наприклад, поверхнями Безьє), які називають гранями. Воно дуже поширене в комп'ютерній графіці, а також використовується для чисельного дослідження оболонкових конструкцій. Проте, ручна побудова таких моделей є досить трудомісткою.

Аналіз наведених вище «класичних» подань показав, що їх практичне застосування обмежене або вимагає значних зусиль для побудови моделей складних об'єктів складної форми. З точки зору універсальності одним з найбільш перспективних виглядає функціональне подання, яке ґрунтується на використанні мови неявних математичних функцій [61–80]. Проте, його використання інженерами вельми важке з точки зору побудови та верифікації моделі.

Одним з поширених способів опису моделей об'єктів зі складною формою в САПР машинобудування є використання проблемно-орієнтованих мов. Наприклад, HyperFun [283], PLaSM [83–84], BDRY [85], QMSH [86] тощо [31–36, 87]. Такі мови, зазвичай, дозволяють будувати моделі конструюючи їх з примітивів або шляхом опису поверхонь об'єкта за допомогою полігональних граней.

1.4. Аналіз методів побудови дискретних моделей форм об'єктів у САПР машинобудування та архітектури

Швидкий розвиток комп'ютерних систем і САПР дозволили значно підвищити швидкість чисельного розв'язку задач, пов'язаних з проектуванням. Незважаючи на розвиток безсіткових методів [88–91] в автоматизованому проектуванні для чисельного аналізу найбільш поширеним є МСЕ [92–94]. Його використовують для розв'язання задач різних класів, які включають статику і динаміку. Ключовою для МСЕ є вимога побудови дискретної моделі форми об'єкта – декомпозиції форми об'єкта на множину елементарних фігур (скінченних елементів).

Під дискретною моделлю форми об'єкта (або сіткою, ці терміни використовують синонімічно) розуміють пару множин: множина вершин і множина зв'язків між вершинами (множина елементів). За суттю, побудова дискретної моделі форми об'єкта – це заміна вихідної неперервної області кінцевим об'єднанням найпростіших фігур: трикутників, чотирикутників, тетраедрів, шестигранників тощо. У проектуванні це визначення можна доповнити обчислювальним процесом, що визначено на дискретній моделі форми об'єкта, за результатами роботи якого вносяться оптимізації у цю модель.

За розмірністю простору дискретні моделі форм об'єктів ділять на одномірні, двовимірні і тривимірні. Дискретні моделі бувають структурованими (рис. 1.2, а) та неструктурованими (рис. 1.2, б, в). У структурованих дискретних моделях ребра елементів відповідають координатним напрямкам деякої (у загальному випадку криволінійної) системи координат. Навпаки, у неструктурованих дискретних моделях вибір ребер не має прив'язки до координатних напрямків. У структурованих дискретних моделях найбільшого поширення набули чотирикутники і шестигранники, а в неструктурованих часто використовують трикутники і тетраедри (які спільно називають «симплексами»).

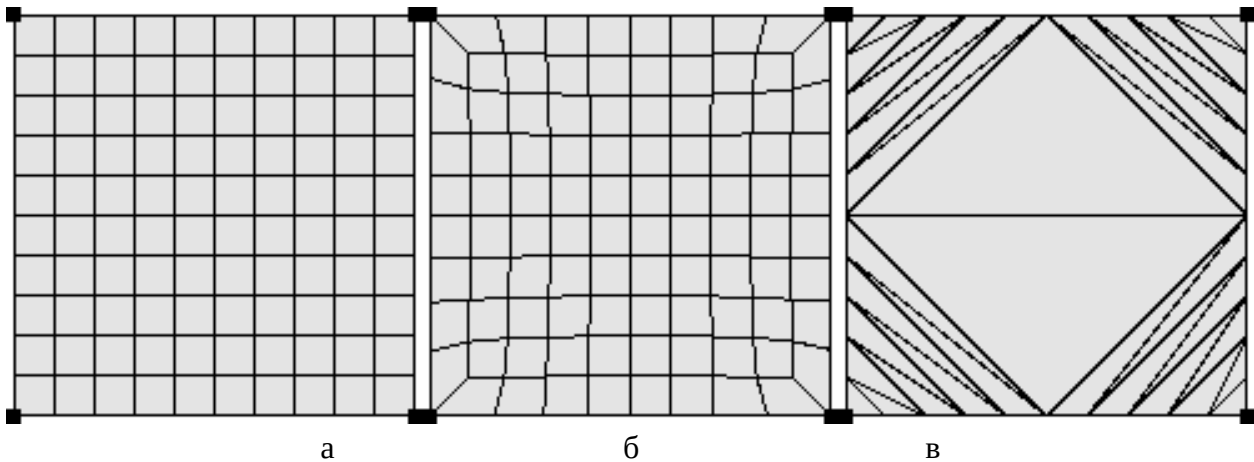


Рисунок 1.2 – Типи дискретних моделей: а – структурована; б – неструктурована (чотирикутники); в – неструктурована (трикутники)

У цілому, найбільш популярними плоскими дискретними елементами є трикутники і чотирикутники. Також у деяких випадках (наприклад, в задачах оптимізації топології об'єктів, дослідження глобальних змін клімату тощо) використовують плоскі шестикутники [95–107]. Аналогічно, у тривимірному просторі найбільш популярними є тетраедри і шестигранники (як природні аналоги, відповідно, трикутника і чотирикутника). Використовують піраміди і трикутні призми при генерації дискретних моделей рухом, а у задачах, наприклад, дослідження сферичних об'єктів розглядають дискретні моделі на базі ікосаедрів [108–112].

1.4.1 Структуровані дискретні моделі форм об'єктів

Основною ознакою структурованих і блок-структурованих дискретних моделей форм об'єктів є наявність регулярних зв'язків між вузлами. Найбільш часто такі моделі використовують чотирикутні елементи на площині або

шестигранні в тривимірному просторі. Водночас для зберігання структурованих дискретних моделей можна використовувати найпростіші структури даних – матриці, в яких сусідні вузли визначають сусідні елементи. Такі моделі виникли як основа методу скінченних різниць.

Структуровані моделі засновано на ідеї трансформації сітки, визначеної на одиничному квадраті (кубі в тривимірному випадку), у форму об'єкта за допомогою відображення $F:[0;1]^n \rightarrow \Omega$ (де n – розмірність простору). Відображення F можна визначити явно (наприклад, як ізопериметричне, конформне тощо) або неявно (наприклад, як розв'язок еліптичних рівнянь з частинними похідними). Актуальні наукові дослідження в області побудови таких дискретних моделей зосереджено на пошуку такого відображення F , яке забезпечить необхідний розподіл вузлів.

При побудові структурованих дискретних моделей зазвичай розглядають дві координатні області: фізичну і розрахункову. Фізична область – це область Ω , для якої будують дискретну модель. Розрахунковою областю виступає одиничний квадрат (куб).

Можна вважати, що у розрахунковій області визначена прямокутна система координат, а у фізичній – криволінійна. Наприклад, якщо фізична область є прямокутником, то побудова розрахункової сітки є вельми тривіальною:

$$\begin{aligned}x_i &= x_0 + \frac{w}{n}(i-1), \quad i = \overline{1, n}, \\y_j &= y_0 + \frac{h}{m}(j-1), \quad j = \overline{1, m},\end{aligned}$$

де $(x_0; y_0)$ – координати нижнього лівого кута прямокутника;

w і h – відповідно, ширина та висота прямокутника;

n – кількість вузлів уздовж осі абсцис;

m – кількість вузлів уздовж осі ординат.

Для криволінійних фізичних областей необхідно знаходити відповідні перетворення координат.

Наприклад, якщо вважати, що фізична область обмежена двома радіусами $r_0 \leq r \leq r_1$ і сторонами кута $0 \leq \alpha \leq \theta$, то перетворення

$$\begin{aligned}x(\xi, \eta) &= [r_0 + (r_1 - r_0)\eta] \cos(\theta \xi), \\y(\xi, \eta) &= [r_0 + (r_1 - r_0)\eta] \sin(\theta \xi)\end{aligned}$$

встановить відповідність з розрахунковою областю у системі координат (ξ, η) (наприклад, якщо $r_0=1$, $r_1=4$ і $\theta=\frac{\pi}{4}$, можна отримати фізичну область у вигляді, поданому на рис. 1.3).

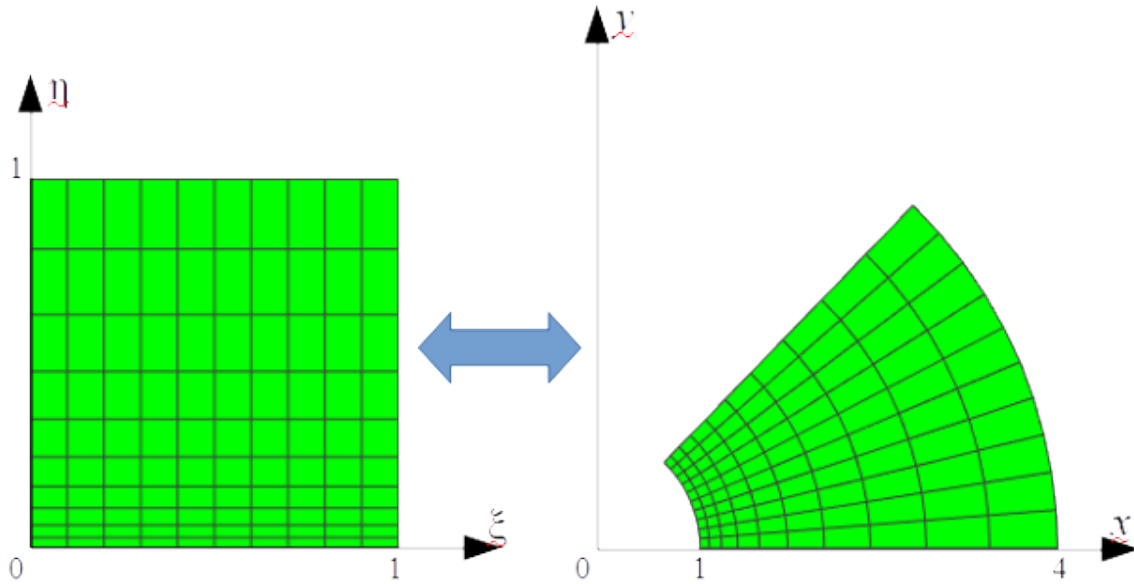


Рисунок 1.3 – Розрахункова та фізична область для кільцевого об'єкта

У випадках, коли подібний аналітичний зв'язок між розрахунковою та фізичною областями встановити важко, використовують методи інтерполяції. Наприклад, якщо область обмежена двома параметричними кривими, то для встановлення відповідності між фізичною та розрахунковою областями, можна використовувати односпрямовану інтерполяцію:

$$\begin{aligned} x(\xi, \eta) &= (1 - \xi)x_l(\eta) + \xi x_r(\eta), \\ y(\xi, \eta) &= (1 - \xi)y_l(\eta) + \xi y_r(\eta), \end{aligned} \quad (1.1)$$

де $\begin{cases} x_l(\eta) \\ y_l(\eta) \end{cases}$ і $\begin{cases} x_r(\eta) \\ y_r(\eta) \end{cases}$ – параметричне подання, відповідно, «лівої» і «правої» кривих (для випадку, коли інтерполяція має горизонтальний характер, рис. 1.4, а).

Якщо інтерполяція має вертикальний характер (рис. 1.4, б), то формули будуть аналогічними до (1.1).

Якщо область обмежена чотирма кривими, то можна визначити відображення, яке ставить у відповідність квадрат $((0;0),(0;1),(1;0),(1;1))$, визначений у розрахунковій системі координат, фізичній області $ABCD$ (рис. 1.5). Таке відображення можна отримати послідовним застосуванням

відображень (1.1) у кожному координатному напрямі з відніманням результатів білінійної інтерполяції:

$$\begin{aligned} x(\xi, \eta) = & (1-\xi)x_l(\eta) + \xi x_r(\eta) + (1-\eta)x_b(\xi) + \eta x_t(\xi) - \\ & -(1-\xi)(1-\eta)x_b(0) - (1-\xi)\eta x_t(0) - \xi(1-\eta)x_b(1) - \xi\eta x_t(1), \\ y(\xi, \eta) = & (1-\xi)y_l(\eta) + \xi y_r(\eta) + (1-\eta)y_b(\xi) + \eta y_t(\xi) - \\ & -(1-\xi)(1-\eta)y_b(0) - (1-\xi)\eta y_t(0) - \xi(1-\eta)y_b(1) - \xi\eta y_t(1). \end{aligned} \quad (1.2)$$

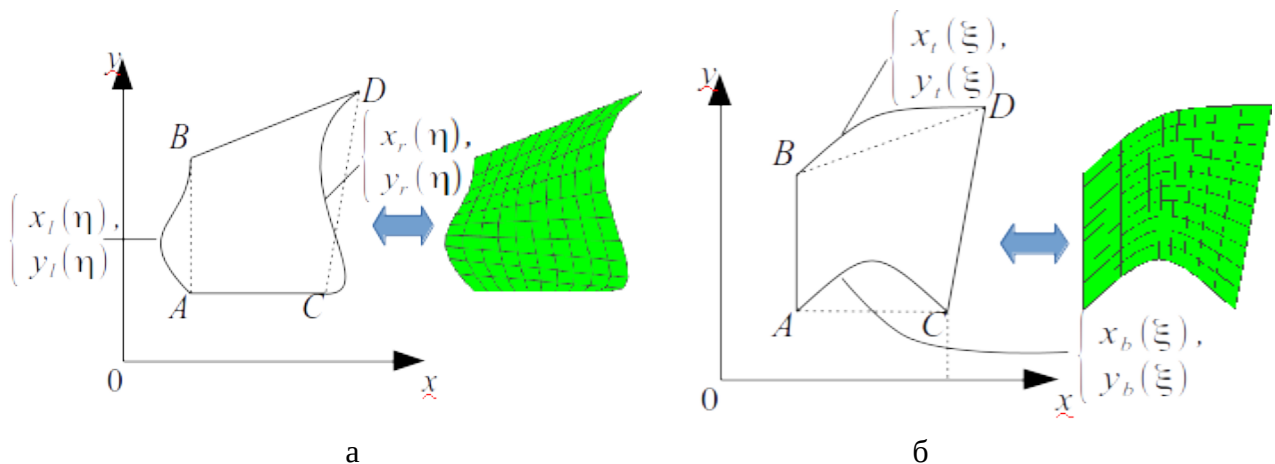


Рисунок 1.4 – Односпрямована інтерполяція: а – горизонтальна; б – вертикальна

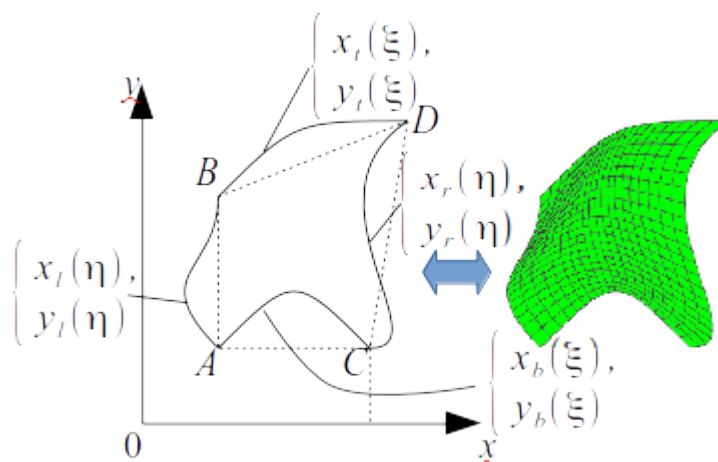


Рисунок 1.5 – Білінійна трансфінитна інтерполяція

Формули, аналогічні (1.1) і (1.2), можна отримати для інтерполяції тривимірних фізичних областей як на основі трилінійної інтерполяції, так і на основі інтерполяції Лагранжа або Ерміта [113]. Методи, що ґрунтуються подібних перетворень, називають алгебраїчними. Один з перших таких методів, що використовує узагальнені криволінійні координати, був запропонований в роботі [114]. Аналогічні формули для побудови тривимірних структурованих дискретних моделей на базі трансфінитної інтерполяції розглянуті в роботах [115, 116–117].

Для контролю за розподілом внутрішніх вузлів, при побудові дискретних моделей на основі алгебраїчних методів, використовують різні функції розтягування [118] (наприклад, експоненціальні, гіперболічні тощо [113]), а також підходи, що ґрунтуються на методі багатьох поверхонь [113, 119], в яких вводяться додаткові поверхні (у двовимірному випадку – криві) для побудови інтерполяційних схем більш високого порядку.

У роботах [120–122, 123, 124] запропоновані алгебраїчні методи, які ґрунтуються на використанні барицентричних координат, для побудови та оптимізації дискретних моделей. Такі схеми переважно використовують для розбиття на скінченні елементи областей, визначених багатокутниками (наприклад, рис. 1.6) або багатогранниками. Для дискретизації областей, визначених неопуклими багатокутниками, розробляють апарат узагальнених барицентричних координат [123].

Основними недоліками алгебраїчних методів є (у загальному випадку) відсутність ортогональності ребер і складність контролю за розміром сусідніх елементів. Ці факти негативно впливають на точність розв’язків, отриманих з використанням таких дискретних моделей.

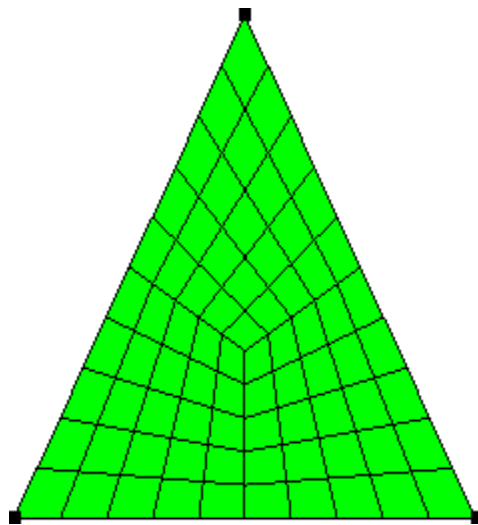


Рисунок 1.6 – Приклад розбиття трикутної області на базі барицентричних координат

Альтернативою алгебраїчних методів для побудови структурованих дискретних моделей є підходи, які ґрунтуються на розв’язанні крайової задачі з частинними похідними. Такі методи називають диференціальними. Їх у залежності від типу базової системи диференціальних рівнянь можна поділити на еліптичні, параболічні та гіперболічні.

Один з найбільш загальних методів побудови дискретних моделей з використанням диференціальних рівнянь з частинними похідними ґрунтуються на розв’язанні рівняння Пуассона [77–80, 119, 113], що мають такий вигляд:

$$\begin{aligned}\frac{\partial^2 \xi}{\partial x^2} + \frac{\partial^2 \xi}{\partial y^2} &= P(\xi, \eta), \\ \frac{\partial^2 \eta}{\partial x^2} + \frac{\partial^2 \eta}{\partial y^2} &= Q(\xi, \eta),\end{aligned}\quad (1.3)$$

де P і Q – функції, які використовують для контролю за щільністю вузлів дискретної моделі.

Розв'язок рівняння (1.3) знаходять у розрахунковій області системи координат (ξ, η) , тому їх можна перетворити до вигляду [119]

$$\begin{aligned}g_{22} \frac{\partial^2 x}{\partial \xi^2} - 2g_{12} \frac{\partial^2 x}{\partial \xi \partial \eta} + g_{11} \frac{\partial^2 x}{\partial \eta^2} + g \left(P \frac{\partial x}{\partial \xi} + Q \frac{\partial x}{\partial \eta} \right) &= 0, \\ g_{22} \frac{\partial^2 y}{\partial \xi^2} - 2g_{12} \frac{\partial^2 y}{\partial \xi \partial \eta} + g_{11} \frac{\partial^2 y}{\partial \eta^2} + g \left(P \frac{\partial y}{\partial \xi} + Q \frac{\partial y}{\partial \eta} \right) &= 0,\end{aligned}\quad (1.4)$$

де

$$\begin{aligned}g_{11} &= \left(\frac{\partial x}{\partial \xi} \right)^2 + \left(\frac{\partial y}{\partial \xi} \right)^2, \\ g_{12} &= \frac{\partial x}{\partial \xi} \frac{\partial x}{\partial \eta} + \frac{\partial y}{\partial \xi} \frac{\partial y}{\partial \eta}, \\ g_{22} &= \left(\frac{\partial x}{\partial \eta} \right)^2 + \left(\frac{\partial y}{\partial \eta} \right)^2, \\ g &= \left(\frac{\partial x}{\partial \xi} \frac{\partial y}{\partial \eta} - \frac{\partial y}{\partial \xi} \frac{\partial x}{\partial \eta} \right)^2.\end{aligned}$$

Рівняння (1.4) розв'язують чисельно, зазвичай, з використанням скінченно-різницевої схеми з крайовими умовами, що відповідають значенням координат вузлів на межі фізичної області. Дискретні моделі, отримані на основі таких систем диференціальних рівнянь, називають еліптичними. Також можна будувати структуровані моделі шляхом розв'язання гіперболічних або параболічних систем [113]. Аналогічні рівняння можна записати для тривимірних тіл (початковими даними будуть положення вузлів на граничних поверхнях) [77–80, 113]. Водночас для нерівномірного початкового розподілу вузлів на межі фізичної області часто використовують алгебраїчні методи для дискретизації кривих [125] і/або двовимірні диференціальні рівняння для дискретизації поверхонь [113].

Однією з найбільш істотних переваг використання рівнянь з частинними похідними є можливість забезпечення ортогональності ліній сітки до межі

області [113], збіжності з лініями течії еквівалентної задачі потенційного обтікання [119], а також гладкості внутрішньої частини сітки при наявності розривів меж. У результаті, методи генерації дискретних моделей, які ґрунтуються на рішенні диференціальних рівнянь, набули поширення у задачах дослідження динаміки газів і рідин [8, 77–80, 119].

Описані вище методи застосовні для кубоподібних тіл (у топологічному сенсі). У разі, якщо тіло має форму, яку неможливо відобразити на куб, то використовують методи блокової декомпозиції (рис. 1.7). Основна ідея таких методів – поділ вихідної геометрії об'єкта на кубоподібні області, до кожної з яких застосовуються методи генерації структурованих дискретних моделей.

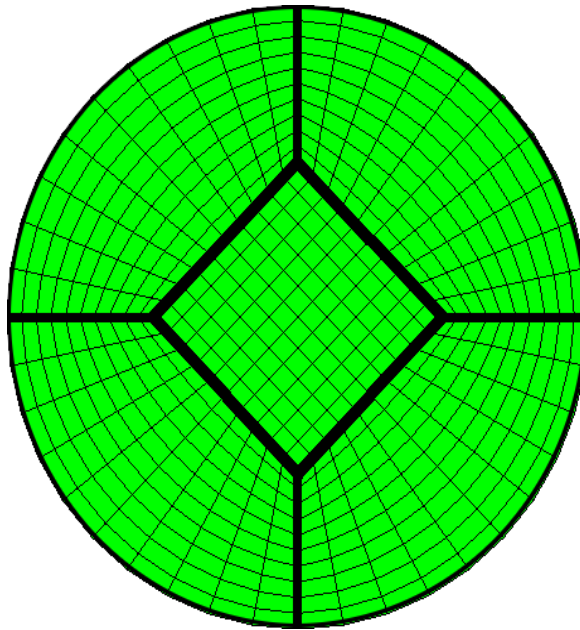


Рисунок 1.7 – Блок-структурована дискретна модель

Один із перших методів блокової декомпозиції ґрунтуються на використанні октодерев і шаблонів для адаптації блоків [126]. Область рекурсивно розбивають на кубоподібні блоки, які в літературі часто називають октантами. Ітерацію по глибині рекурсії виконують, як правило, або в околі особливої точки, або межі з високою кривиною (наприклад, рис. 1.8). На заключному етапі за допомогою спеціальних шаблонів відновлюють коректність топології. Потім будують структуровані сітки для кожного з блоків. Розвиток цієї ідеї висвітлено у [127–129], а узагальнення на тривимірний випадок – у [130–131].

Іншим поширеним способом декомпозиції на блоки є використання методу серединних осей [132–134]. Серединні осі пов'язані з поняттям «скелет багатокутника» – геометричного місця точок кривої, що утворена центрами вписаних кіл максимального радіуса, який дотикається до меж об'єкта. Вписане

коло повинне дотикатись, щонайменше, двох точок межі об'єкта (приклад наведено на рис. 1.9). Інакше, повинно існувати інше вписане коло, що повністю задовольняє цим вимогам. Подання області у вигляді серединних осей є вихідним, розбиття на блоки проводиться в області зламу серединної осі. Цей підхід має узагальнення для тривимірного випадку під назвою серединної поверхні. Варто зазначити, що побудова серединних осей або поверхонь є самостійною досить складною задачею, методи вирішення якої часто використовують триангуляцію Делоне або діаграми Воронова.

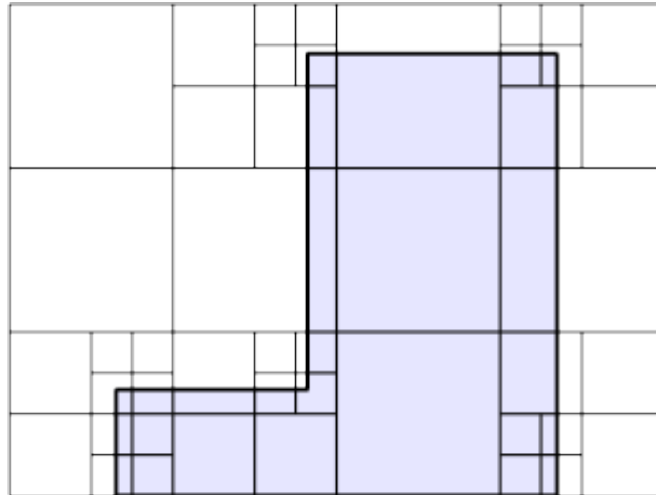


Рисунок 1.8 – Загальна ідея рекурсивного розбиття на блоки

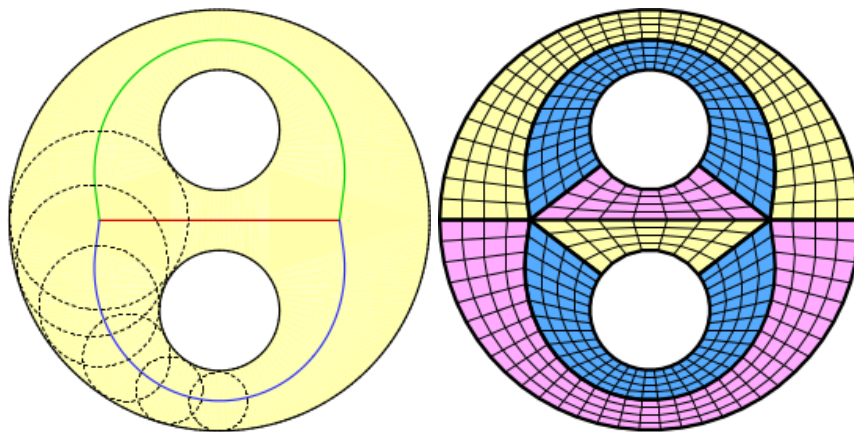


Рисунок 1.9 – Серединні осі: початкова область і побудовані на ній блоки [132]

Розробляють методи декомпозиції області на блоки на основі нечіткої логіки [135–137], методи на основі загрублення (coarsening) вихідної сітки [138] тощо [139, 140]. Алгоритми на основі нечіткої логіки є вельми перспективними, однак їх практичне застосування ускладнюється труднощами їх реалізації. Методи на основі загрублення як вихідні дані використовують початкову досить густу дискретну модель, в якій блоки формуються шляхом об'єднання сусідніх елементів, що дозволяє їх використовувати для перестроювання моделі за результатами розрахунку.

Однією з переваг блокової декомпозиції є можливість використання технологій паралельних обчислень при обробці блоків (особливості реалізації таких методів розглянуті в роботах [141–143]).

Отже, можна зазначити ряд переваг і недоліків використання структурованих дискретних моделей форм об'єктів. До переваг можна віднести можливість використання більш високого порядку апроксимації (наприклад, у задачах газодинаміки і дослідження переміщення рідин і газів), можливість використання меншої кількості обчислювальних ресурсів (не потрібно зберігати інформації про сусідні вершини, ребра, грані тощо). Проте, адаптація структурованих або блок-структурованих моделей до складних тіл вельми трудомістка і водночас можлива поява вироджених елементів. Декомпозицію на блоки можна застосовувати при використанні методів твердотілої геометрії для подання форми об'єктів (блоки та їх границі подані явно). Однак, в разі тіл складної форми або при функціональному поданні методи декомпозиції на блоки практично не застосовні (через слабку формалізації блоків або відсутності явного подання границь).

1.4.2 Неструктуровані дискретні моделі форм об'єктів

Методи побудови неструктурованих дискретних моделей можна розділити на дві підгрупи:

- 1) методи побудови дискретних моделей на базі трикутників і тетраедрів (симплекси);
- 2) методи побудови дискретних моделей на базі чотирикутників і шестигранників (топологічні кубики).

Симплекси набули широкого поширення завдяки їх топологічній гнучкості (для них визначені прості, добре формалізовані процедури вставки або видалення вузла, що дозволяє порівняно легко згущувати або загрублювати моделі).

Використання чотирикутників і шестигранників дозволяє більш точно моделювати об'єкт, що проектується, при використанні меншої кількості вузлів. Водночас генерація дискретних моделей на базі чотирикутних або шестигранних елементів є більш складною у порівнянні з трикутниками або тетраедрами. Це можна проілюструвати на такому прикладі. Нехай дана структурована сітка, в яку необхідно додати вузол, використовуючи тільки локальні перебудування. Хоч це і можливо в плоскому випадку (рис. 1.10), але у тривимірному просторі не є можливим [113]. Отже, техніка, яку активно

використовують для симплексів у методах на основі критерію Делоне [26], практично не застосовна для чотирикутників і шестигранників.

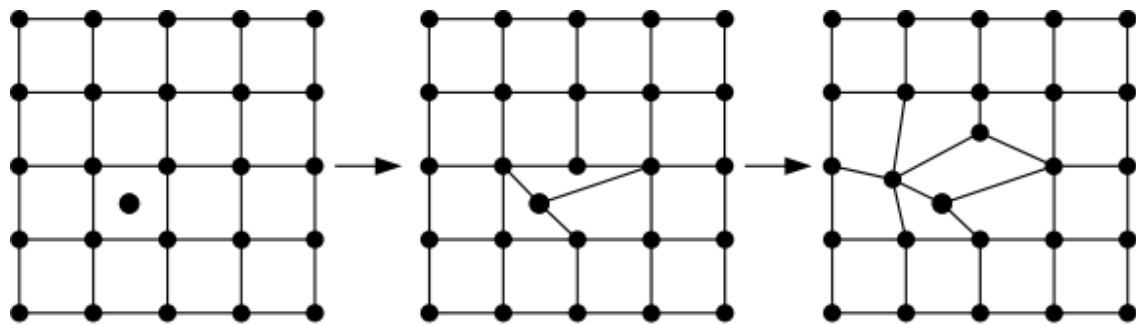


Рисунок 1.10 – Додавання вузла до сітки чотирикутних елементів

1. Методи генерації дискретних моделей на базі трикутників і тетраедрів

Однією з найбільш поширених на практиці є проблема побудови триангуляції вихідної області, придатної для використання в методах скінченних або граничних елементів. Проблема полягає в знаходженні множини трикутників, що покривають вихідну область, форма і розміри яких задовольняють наступним вимогам: не повинно бути трикутників з дуже маленькими і дуже великими кутами, трикутники не повинні бути меншими, ніж це необхідно, і не повинні перевищувати задані розміри. Методи на основі критерію Делоне [144] математично гарантують, що такі вимоги можуть бути задоволені. Такі ж вимоги, зазвичай, висуваються й до дискретних моделей на базі тетраедрів, для яких математичним обґрунтуванням слугує узагальнення критерію Делоне на випадок n -мірного простору [145–146].

Сьогодні досить добре розвинені методи побудови триангуляції Делоне [144] для вихідної множини точок на площині, детальний аналіз яких можна знайти в роботах О. В. Скворцова [147–149], а також Р. Су і Р. Л. С. Драйсдейла [150]. Розроблені методи від найпростіших – інкрементних алгоритмів, які додають нові вузли в модель один за іншим, до найбільш швидких, які засновані на принципі «розділяй і володарюй» [150]. Вони відносно легко узагальнюються на тривимірний випадок [152–155] (достатньо змінити процедуру перевірки умови Делоне [156]). Ці методи отримали своє застосування в задачах комп'ютерної графіки та геоінформаційних системах. Проте, в задачах проектування потрібна апроксимація геометричних меж об'єкта ребрами і гранями елементів, що не враховується в цих методах.

Також розроблені способи та методи побудови триангуляції Делоне для областей, поданих багатокутниками або планарними графами [157–163]. У цій групі методів найбільшого поширення набули модифікації методів Рапперта (J. Ruppert) [162] та Чю (L. Chew) [160–161], які є способами підвищення якості сітки, згенерованої для вихідної області на базі критерію Делоне. Їх

узагальненням для тривимірного простору є методи побудови покриття тетраедрами області, обмеженої плоскими гранями [164–166]. В основі таких методів, зазвичай, знаходиться ітераційне додавання нового вузла для локального поліпшення геометричних характеристик дискретної моделі. Новий вузол може додаватися в дискретну модель для усунення елемента з невідповідною якістю в центр описаного кола (рис. 1.11) для трикутника [167] (описаної сфери для тетраедра), або в його цент мас [171]. У деяких роботах [168–169] використовують розбиття Воронова як проміжне для побудови триангуляції Делоне.

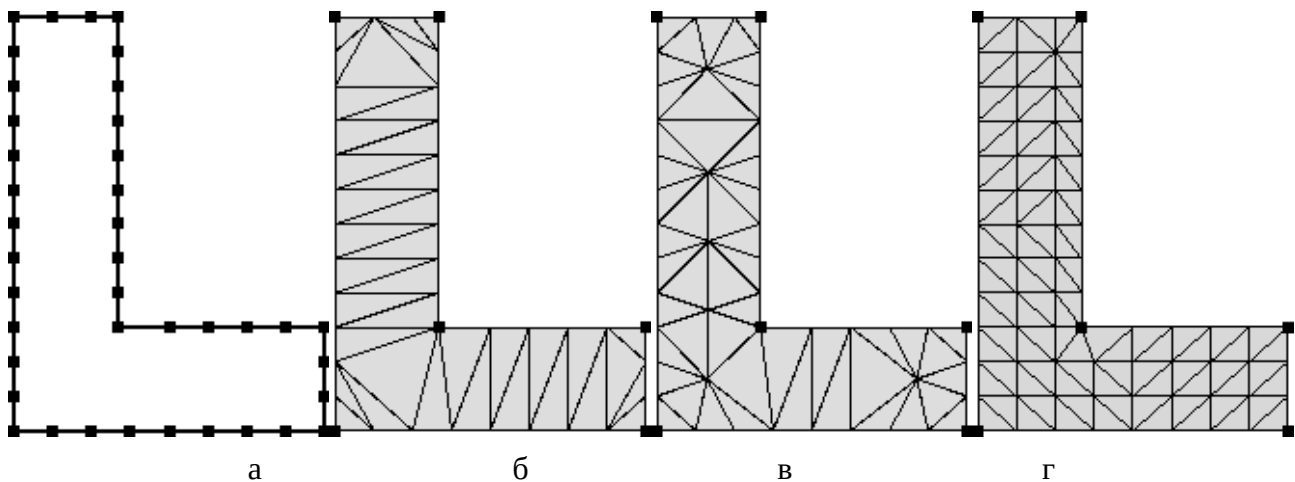


Рисунок 1.11 – Триангуляція L-області: а – вихідний планарний граф; б – триангуляція Делоне; в – оптимізація мінімального кута й площі додаванням вузлів у центр описаного кола; г – дискретна модель отримана з використанням техніки граничної корекції

Техніки, засновані на додаванні нового вузла у центр кола, описаного навколо трикутника, не гарантують, що всі ребра та грані вихідного багатогранника будуть апроксимовані, відповідно, ребрами або гранями елементів. Для вирішення цієї проблеми використовують техніки додавання нових вузлів в середини відсутніх ребер або граней [162, 170–172], які іноді комбінують з процедурою дзеркального відображення (flip) [173–172]. Проте, в задачах проектування як плоских, так і тривимірних технічних об'єктів складної форми подання їх меж у вигляді планарних графів або плоских граней є досить складною та нетривіальною задачею, що ускладнює практичне використання таких методів.

Методи рухомого фронту [174–182] ґрунтуються на ідеї пошарового заповнення області елементами. Вихідною інформацією для них є дискретна модель межі області, яка утворює початковий фронт. Рухом фронту формують шари елементів до тих пір, поки вихідна область не буде заповнена повністю (приклад наведено на рис. 1.12). Фронтальні методи запропоновані для генерації моделей як плоских (на базі трикутників) [174, 179–182], так і тривимірних (на

базі тетраедрів) [175–176, 178] об'єктів. У деяких роботах комбінують використання методу на основі критерію Делоне з рухомим фронтом [177, 180]. Фронтальні методи досить універсальні, проте, необхідно відзначити їх високу ресурсомісткість і відносно низьку швидкість роботи. Їх недоліком є проблема самоперетину фронтів. Також отримання якісного початкового фронту (особливо в тривимірному випадку) є самостійною складною задачею.

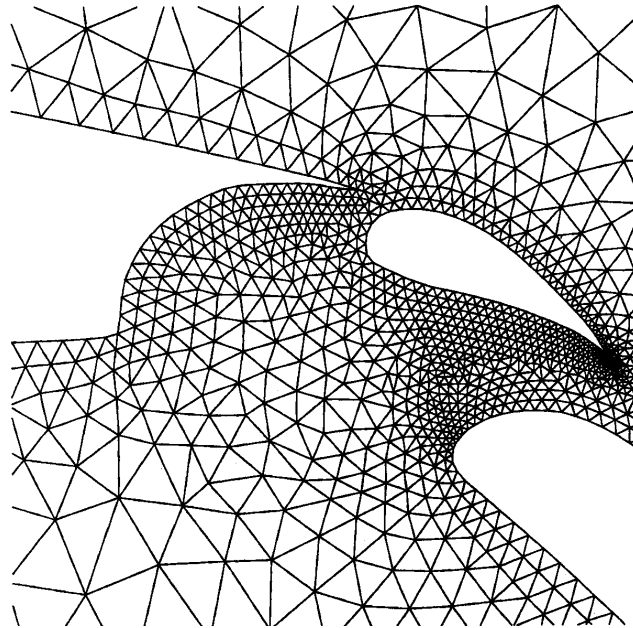


Рисунок 1.12 – Приклад частини фронтальної триангуляції [181]

Інша група методів генерації дискретних моделей базується на використанні деякої початкової сітки з адаптацією її межових вузлів до меж області [158, 165, 183–186, 189]. Вихідна сітка, зазвичай, є відносно простою (наприклад, структурована сітка). За такої умови накладається єдина вимога: розмір комірок повинен бути меншим, ніж мінімальна геометрична особливість області. На етапі адаптації тут поширене використання граничної корекції вузлів [183] (приклад застосування цього підходу наведено на рис. 1.11, г). Вперше ідея використання вихідної сітки на базі квадратних елементів для рівномірної триангуляції двовимірної області, обмеженою планарним графом, була висвітлена у роботі [184] (доведено, що мінімальний кут у трикутниках буде не меншим 13° , якщо у вихідному графі немає кутів, гостріших 13°). Для побудови нерівномірних дискретних моделей використовуються адаптивні вихідні сітки на базі методу Quadtree [126, 129, 187, 188]. У [158] показано, що при використанні адаптивної вихідної сітки кути в елементах (виключаючи гострі кути між ребрами на межі) будуть більшими за $18,4^\circ$. Цей результат був поліпшений в роботі [189], у якій доведено, якщо в межі області, поданої планарним графом, відсутні гострі кути, то у результаті триангуляції всі кути

елементів будуть знаходитися в межах від 30° до 90° . Даний підхід відносно просто узагальнюється на випадок тривимірного простору [185–186, 188]. Варто відзначити, що зазначені вище роботи використовують дискретну модель межі об'єкта як вхідні дані. Задача її отримання є складною, тут існує можливість отримання вироджених елементів біля межі області.

2. Методи на генерації дискретних моделей на базі чотирикутників і шестигранників

Прямі фронтальні методи генерації неструктурованих сіток чотирикутних елементів [190–196] будують модель високої якості, адаптовану до границь з малою кількістю нерегулярних елементів. Їх основна ідея полягає у використанні початкової дискретизації межі та подальшому заповненні вихідної області скінченними елементами шар за шаром (рис. 1.13). Прямі фронтальні методи також узагальнені на випадок тривимірного простору для отримання неструктурованих сіток шестигранних елементів. В такому випадку область вичерпують пласт за пластом, формуючи таким чином і дискретизацію фронту у вигляді чотирикутних граней, і дискретизацію внутрішньої частини області [194, 197–198]. Основними недоліками прямих фронтальних методів, що призводять до складності їх практичної реалізації та великих обчислювальних витрат, є: 1) трудомісткість пошуку вироджених елементів при перетині двох фронтів; 2) складність запобігання виникненню пустот (ділянок області, незаповнених елементами) [113]. Одночасно, отримання якісної початкової дискретизації меж об'єкта є досить складною (особливо в тривимірному випадку) самостійною задачею.

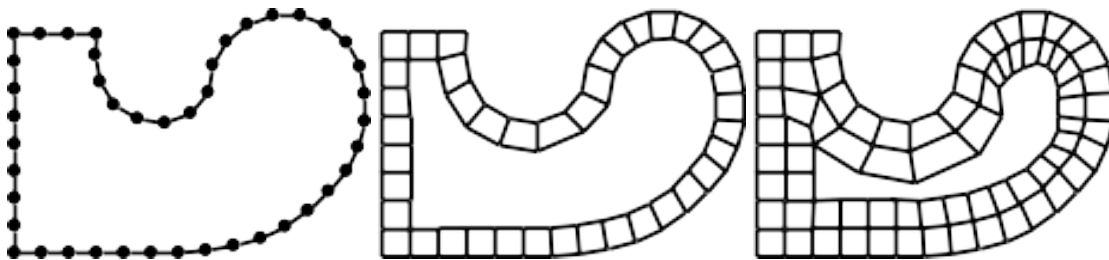


Рисунок 1.13 – Приклад етапів фронтального метода

Непрямі фронтальні методи використовують попередньо побудовану сітку трикутників або тетраедрів. Їх основна ідея – це пошарова трансформація трикутників у чотирикутники [199–203] або тетраедрів у шестигранники [204–205]. На відміну від прямих фронтальних методів для них відсутня проблема перетину фронтів, вони захищені від появи порожнеч усередині контуру, але найчастіше вони будують сітку топологічно менш якісну, ніж кращі прямі фронтальні методи [199–200]. Серед непрямих фронтальних методів можна виокремити метод Q-Morph, запропонований Стівеном Оуеном (S. Owen) [202–

203], та його модифікацію, розроблену А. В. Сковпенем [199–200], яка будує сітку, близьку до оптимальної, з використанням трансформацій трикутників. В результаті узагальнення методу Q-Morph на тривимірний випадок запропонований метод H-Morph [204–205]. Проте, недоліком даного підходу є необхідність наявності якісної попередньої дискретизації на базі трикутників або тетраедрів, отримання якої є окремою трудомісткою задачею.

Техніка просторового вигину континууму (the Spatial Twist Continuum), описана у роботах [206–208], дозволяє автоматично або напівавтоматично генерувати сітки чотирикутних і шестигранних елементів. В основу техніки генерації дискретних моделей закладено STC-подання, яке є двоїстим для чотирикутників (рис. 1.14) або шестигранників (рис. 1.15) [207–208], що робить його придатним для аналізу якості сіток. STC-подання, по суті, є аналогом діаграм Воронова. Воно будується для дискретної моделі межі об'єкта і потім конвертується в дискретну модель на базі чотирикутників або шестигранників. Основним недоліком алгоритмів, що використовують STC-подання, є висока обчислювальна складність: у гіршому випадку вони стають алгоритмічно повними [113]. У деяких роботах [209–211] пропонується використовувати метод малих бічних зсувів (Whisker Weaving), який будує STC-подання на базі рухомого фронту. Недоліки такого підходу аналогічні недолікам прямих фронтальних методів. Загалом, необхідно відзначити, що результатом побудови двоїстого подання в тривимірному випадку не завжди є валідна сітка шестигранних елементів. Також досить важко реалізувати стійку версію алгоритму для методу на основі STC-подання.

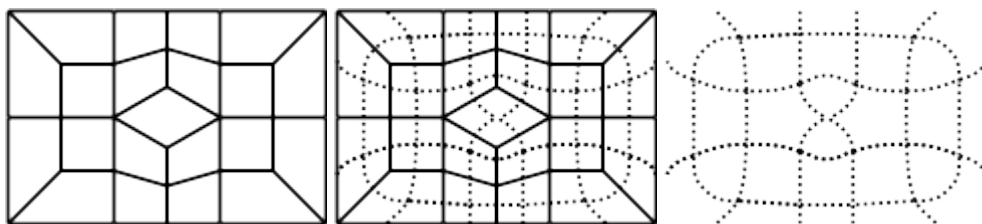


Рисунок 1.14 – Сітка чотирикутних елементів та її STC-подання

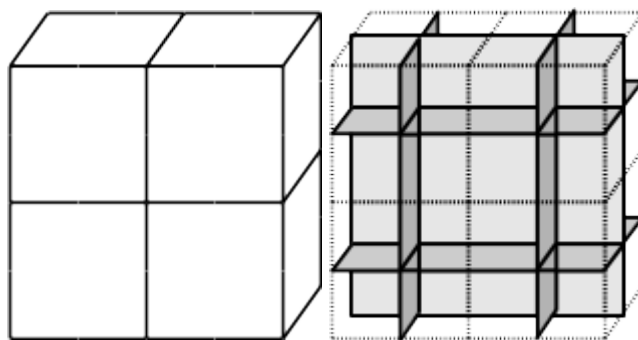


Рисунок 1.15 – STC-подання сітки шестигранних елементів

Методи суперпозиції ґрунтуються на використанні деякої базової стратегії для побудови дискретних моделей на базі чотирикутників або шестигранників. Алгоритми даного класу, зазвичай, як вихідні дані використовують деяку вихідну сітку, яка більш-менш просто може бути побудована і покриває область навколо геометричного об'єкта. Потім вихідну дискретну модель адаптують до його меж (рис. 1.16). Необхідно відзначити, що ключовим кроком даної групи методів є складність генерації сітки високої якості біля межі об'єкта (межа має бути апроксимована ребрами елементів, які повинні бути невідродженими), а їх практична реалізація досить проблематична [113]. Існує два основні підходи до адаптації вихідної сітки до межі об'єкта: сітковий [212–214] і проєкційний [215, 216]. Перший підхід ґрунтується на ідеї пошуку точок, розташованих на межі області, відповідних граничним точкам початкової сітки, і подальшому формуванні шару прикордонних елементів за допомогою техніки ізоморфізму [212]. Другий підхід ґрунтується на переміщенні прилеглих вузлів вихідної сітки на межу об'єкта. Також методи на основі суперпозиції можна використовувати для адаптації нерівномірною початкової сітки. Для цього, як і в методах фонові сітки для симплексів, використовують октодерева [128, 130, 213–214, 219]. Спільною особливістю методів на основі суперпозиції є генерація елементів приблизно однакового розміру [113]. Водночас в результаті задоволення вимоги, щодо співвідношення розмірів елементу вихідної сітки та найменшої геометричної особливості, отримана дискретна модель може бути надмірно густою.

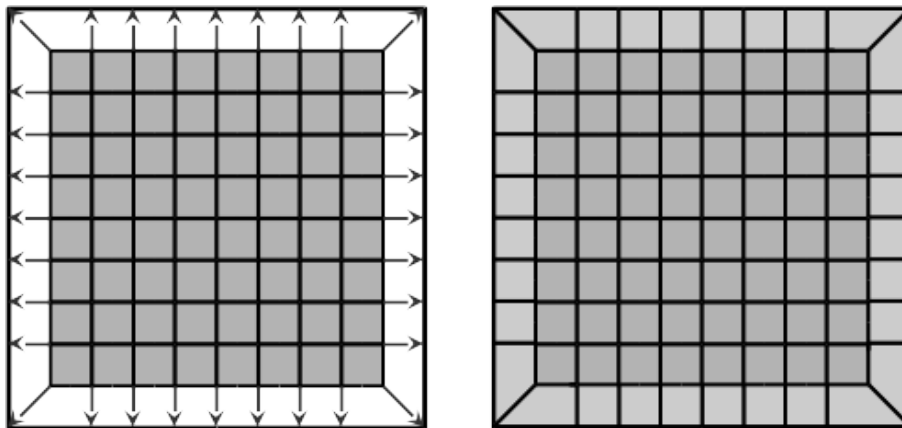


Рисунок 1.16 – Дискретизація квадратної області з використанням фонові сітки

1.4.3 Дискретні моделі оболонкових конструкцій

Для моделювання багатьох об'єктів машинобудування (наприклад, елементів ракетоносіїв, літаків, морських суден, автомобілів тощо) використовують оболонки. Оболонка – форма тіла, у якого один з розмірів

(товщина) значно менше інших. Геометрично оболонку, зазвичай, подають серединною поверхнею. Серединні поверхні, зазвичай, задають за допомогою параметричних або неявних функцій.

Дискретні моделі оболонкових об'єктів машинобудування, зазвичай, використовують трикутні або чотирикутні оболонкові елементи. Такий підхід має кілька переваг відносно використання тривимірних елементів (солідів: тетраедрів або шестигранників): по-перше, економія обчислювальних ресурсів за рахунок використання меншої кількості елементів, по-друге, при використанні тонких тривимірних елементів виникають великі коефіцієнти в матриці жорсткості [220] в позиціях напрямів, відповідних товщині оболонки, що негативно впливає на точність моделювання.

Важливим є той факт, що частина методів побудови дискретних моделей на базі тетраедрів [175–176, 178] або шестигранників [194, 197–198] використовують дискретні моделі межі на базі, відповідно, трикутників або чотирикутників як початкові.

При параметричному поданні форм оболонок поширена ідея побудови множин вузлів та елементів в двовимірному параметричному просторі з подальшим відображенням на поверхню. Проте, якісна дискретна модель, отримана в двовимірному параметричному просторі, необов'язково буде відповідати якісній моделі, що складається з елементів на поверхні.

1. Методи побудови дискретних моделей поверхонь з використанням трикутних елементів

У роботах [221–222] розглянуті особливості використання відображень для побудови дискретних моделей з використанням трикутників для повністю складених з лінійних і квадратичних форм або B-сплайнів параметричних поверхонь. Для покращення якості дискретних моделей запропоновані техніки перестановки вузлів (swar) і згладжування Лапласа.

Також розроблені методи, які ґрунтуються на використанні рухомого фронту [223–226]. Як і в двовимірному випадку, вихідними даними є дискретна модель меж (ребер) поверхні, внутрішня частина якої заповнюється шар за шаром. У роботах [224–226] запропоновані метрики для управління процесом побудови фронту та схема використання фонові сітки на базі методу Quadtree для поліпшення якості дискретної моделі. Схожа з рухомим фронтом ідея заповнення поверхні «біжучими» (marching) трикутниками з попереднім перетворенням подання поверхні з параметричного в неявне розглянута в [227]. Недоліки методів на основі рухомого фронту для поверхонь такі самі як у відповідних методів для суцільних областей.

Ряд робіт ґрунтується на ідеї адаптації методу Рапперта [162] для тріангуляції поверхні на базі критерію Делоне [144]. Ключовим тут є спосіб перевірки виконання умови потрапляння в описане коло для тривимірного трикутника та відображення його центру в параметричний простір. Одна з перших таких робіт [228] заснована на безпосередній перевірці виконання попадання в описане коло у 3D. Така перевірка вельми екстенсивна з обчислювальної точки зору, тому часто використовують різні схеми апроксимації. Наприклад, у [229] запропонована схема використання основних кривин для апроксимації описаного кола поверхневого трикутника двовимірним еліпсом в параметричному просторі; у [230] запропонований підхід, заснований на використанні сфер.

Альтернативним параметричному поданню поверхонь, зазвичай, є неявне. Методи побудови тріангуляції поверхонь, визначених неявними функціями, можна умовно розділити на три групи: коміркові [231], біжучих трикутників [232] і методи на основі тесселяції [233].

2. Методи побудови дискретних моделей поверхонь з використанням чотирикутних елементів

Існують два основні підходи до генерації дискретних моделей на базі чотирикутників для оболонкових конструкцій: прямий і непрямий. Як і в двовимірному випадку, прямі схеми відразу заповнюють область чотирикутниками, а непрямі використовують попередньо побудовану дискретну модель на базі трикутників.

Історично першими прямими до дискретизації поверхні чотирикутниками є техніки на базі відображень [234–237]. Елементи генерують для фізичної області шляхом відображення рівномірної або близькою до рівномірної сітки, побудованої в параметричній площині, на криволінійну поверхню. Такі техніки дозволяють отримати структуровані дискретні моделі високої якості. Однак, застосовні вони для відносно простих областей. Складні багатозв'язні області повинні бути розбиті на блоки [238]. Проте, можливості автоматичного застосування таких технік залишається вельми обмеженими, водночас вони потребують значних обчислювальних ресурсів.

Альтернативна техніка генерації дискретних моделей криволінійних поверхонь з використанням чотирикутників заснована на методі рекурсивної декомпозиції [239]. Також розроблені прямі фронтальні методи (наприклад, [192]), які будують модель поверхні шар за шаром.

У непрямих методах чотирикутники формують, конвертуючи елементи попередньо побудованої моделі на базі трикутників за допомогою операцій злиття або розщеплення. Найпростіша схема для побудови множини

чотирикутників (без гарантій повної відсутності трикутників у результаті), шляхом видалення діагоналей між суміжними трикутниками запропонована в [240]. Вона була розвинута в [241] до гарантованої відсутності трикутників у результаті для довільної області. Аналогічний підхід до конвертації трикутників у чотирикутники використаний у [242]. Техніка одночасної генерації трикутників та їх злиття у чотирикутники запропонована у [190]. З огляду на те, що непрямі техніки використовують вихідну сітку трикутників, порівняно нескладно забезпечити нерівномірну щільність вузлів моделі (наприклад, [223, 243]). Проте, результат роботи непрямих методів – менш якісні дискретні моделі (відносно кращих прямих).

1.5. Аналіз методів оптимізації дискретних моделей форм об'єктів

Точність результатів, яку дозволяють отримати САПР машинобудування, у тому числі, залежить від якості дискретних моделей форм об'єктів. При розробці методів поліпшення якості дискретних моделей найбільш часто використовуються дві характеристики: якість форми елемента (близькість до правильної форми, тобто, відсутність дуже гострих кутів, співвідношення довжин сторін близьке до одиниці тощо) та/або близькість сітки до рівномірної.

1.5.1 Методи оптимізації внутрішніх вузлів та елементів

Найбільшого поширення набули підходи, які ґрунтуються на згладжуванні Лапласа [244], що базується на ідеї переміщення оброблюваного вузла у центр мас багатокутника, утвореного сусідніми вузлами (рис. 1.17). Водночас актуальні напрямки досліджень у даній групі методів базуються на введенні системи вагових коефіцієнтів або контрольних функцій, заснованих, наприклад, на значеннях кутів елемента [245], спеціальних інтегральних функцій [246] тощо [190, 192, 247].

Альтернативою згладжування Лапласа є підходи, засновані на введенні цільової функції, оптимізація якої забезпечить найкраще розташування вузлів сітки. Наприклад, як цільові функції можуть виступати значення мінімального або максимального кута елемента [248–250], співвідношення радіуса описаного кола до радіусу вписаного (для трикутників та тетраєдрів) [251], площі елементів, довжини сторін або скалярні добутки векторів, визначених на

сторонах елементів [252–256]. У цій групі можна виділити методи, засновані на вирішенні локальної (подібно згладжування Лапласа переміщається тільки один вузол) [250, 257] і глобальної (мінімізації підлягає функція координат всіх внутрішніх вузлів) [252–256] оптимізаційних задач.

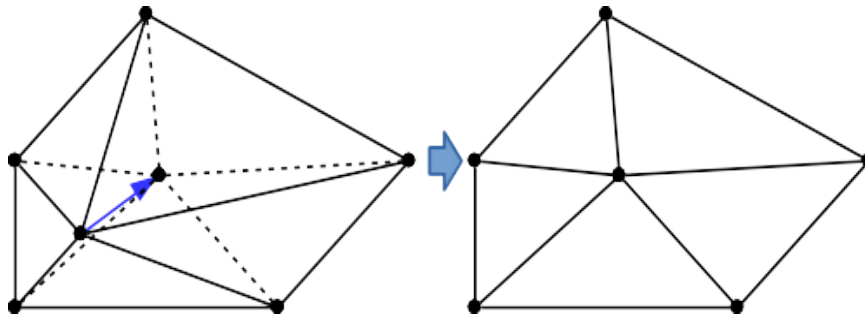


Рисунок 1.17 – Згладжування Лапласа

Перевага методів, які ґрунтуються на оптимізації, полягає в гарантуванні поліпшення якості сітки. Проте, їх реалізація вимагає розв’язання задачі оптимізації з числом змінних вдвічі або втричі більшим, ніж кількість вузлів сітки. Час, витрачений на виконання такої процедури, може у 30–40 разів перевищувати час, необхідний на виконання згладжування Лапласа [258].

Окремо розробляють гібридні методи [248, 249, 259], які ґрунтуються на комбінуванні оптимізаційних методів і згладжування Лапласа.

Описані вище методи, які ґрунтуються на оптимізації геометричних або топологічних характеристик дискретної моделі. Однак, в деяких задачах, наприклад, задачах побудови дискретних моделей для дослідження напружено-деформованого стану пластин, які перебувають під дією навантаження деякої форми, виникає необхідність управління процесом поліпшення сітки. Для підвищення точності інженерного аналізу необхідно оптимізувати також і геометричні характеристики елементів моделей, і розподіл їх щільності у залежності від фізичної постановки. Так, наприклад, у праці [260] розглянута схема оптимальної побудови дискретної моделі для чисельного розв’язання рівняння Пуассона-Больцмана. У [261] розглянута схема оптимізації дискретної моделі на базі чотирикутників з керуванням спеціальною функцією.

1.5.2 Методи оптимізації межових вузлів та елементів

Методи оптимізації положень межових вузлів та елементів мають одночасно найбільші складність та практичний інтерес. Перші праці у цьому напрямку пов’язані з методом крокуючих кубиків (marching cubes) [231], у

якому для швидкої візуалізації поверхонь тривимірних геометричних об'єктів, заданих неявними функціями, будуються трикутні елементи за шаблонами значень у вузлах рівномірної сітки.

Для пошуку особливостей поверхонь тривимірних геометричних об'єктів (наприклад, значних змін кривини) у праці [262] запропонована модифікація методу крокуючих кубів, яка ґрунтується на аналізі градієнтів функції відстані та вставці нових вузлів. У працях [263–264] запропоновані підходи до поліпшення результатів роботи методу крокуючих кубиків з використанням дуальних сіток [263] або метрик [264], що враховують відхилення вузлів від поверхні та нормалей трикутників від градієнтів неявної функції. У праці [265] для оптимізації початкової тріангуляції запропоновано мінімізувати енергетичний функціонал при пошуку положень вузлів та зменшенні їх кількості. У праці [266] спільно з енергетичним функціоналом запропоновано використовувати неявну інтерполяцію поверхні.

Точніше подання особливостей можна отримати за допомогою адаптивних дискретних моделей. Наприклад, у праці [267] запропоновано використовувати ієрархічні структури (octrees) для прискорення роботи методів, заснованих на покроковому скануванні областей, а у працях [224, 268, 269] для збільшення кількості вузлів в областях з найбільшою кривиною поверхні. Ієрархічні структури також можуть використовуватися спільно з аналізом перетинів нормалей з поверхнею [270]. Використання ієрархічних структур для оптимізації дискретних моделей при адаптивному скінченно-елементному аналізі показано в роботі [226]. У праці [271] розглянуті особливості реалізації ієрархічних структур для оптимізації моделей з використанням графічного процесора.

Альтернативою сканування області елементарними об'ємами (кубами або тетраедрами) є сканування поверхні плоскими елементами. Наприклад, в методі крокуючих трикутників [272] поверхню заповнюють трикутниками, почавши рух з довільної її точки. У працях [232, 273] наведені адаптивні схеми для сканування неявних поверхонь трикутниками. Схожі ідеї реалізовані у фронтальному методі [227].

У працях [160, 228, 274] запропоновано використовувати критерій Делоне з обмеженнями при генерації або оптимізації координат трикутників на поверхні. У роботі [275] схема з критерієм Делоне застосована для адаптивної тріангуляції поверхонь з урахуванням матеріалів (складених об'єктів).

У праці [276] запропонований підхід до пошуку особливостей поверхонь, який ґрунтується на сегментації та кластеризації при оптимізації мереж трикутних елементів. Для розв'язання схожої задачі у праці [277] використовують семантичну оптимізацію.

1.6. Подання форм об'єктів у хмарних САПР з використанням неявних функцій

Першим етапом при чисельному дослідженні стану об'єкта машинобудування є опис його форми. Типова задача цього етапу є такою. Дано: креслення або опис об'єкта. Необхідно побудувати математичну модель форми об'єкта. При розв'язанні такої задачі можна виокремити два основних класи форм об'єктів: суцільні області (або тіла) та оболонки. Суцільні тіла – це форми, які можна розглядати як замкнені та обмежені підмножини евклідова простору. Особливістю оболонок є те що, у них один з розмірів (товщина) значно менший за інші. Отже, необхідно враховувати їх особливості.

З математичної точки зору тривимірне тіло – це підмножина тривимірного простору $\mathbb{R}^3$. В контексті автоматизації такі підмножини повинні бути замкнутими, обмеженими та регулярними в $\mathbb{R}^3$. Їх у роботі [45] називають g -множинами. Множини називають регулярними, якщо результат об'єднання, перетину або різниці будь-яких двох регулярних множин є регулярною множиною (порожню множину також вважають регулярною).

Для подання призматичних тіл зі сталою товщиною (наприклад, при дослідженні плоского деформованого стану) використовують двовимірні області, які, за аналогією з тривимірним випадком, моделюються замкнутими, обмеженими і регулярними підмножинами в $\mathbb{R}^2$.

При автоматизації g -множини можна подати явно (перерахуванням всіх точок, що їй належать) або неявно (формуванням умови для перевірки приналежності точки до неї). Явно описати g -множину можна лише в найпростіших випадках, тому неявний опис більш поширений.

Неявно g -множини можна визначити різними способами:

- 1) перерахуванням умов належності точки множині;
- 2) формуванням системи нерівностей або рівнянь, коренями яких збігаються з множиною;

3) формуванням єдиного рівняння або нерівності, корені якого збігаються з множиною. Наприклад, двовимірну квадратну область (рис. 1.18) при першому способі можна описати як область, розташовану між прямими $x = -1$, $y = 1$, $x = 1$ і $y = -1$. Її також можна подати системою нерівностей

$$\begin{cases} 1+x \geq 0, \\ 1-x \geq 0, \\ 1+y \geq 0, \\ 1-y \geq 0, \end{cases}$$

або однією нерівністю $2 - x^2 - y^2 - \sqrt{(1 - x^2)^2 + (1 - y^2)^2} \geq 0$ (рис. 1.19, а).

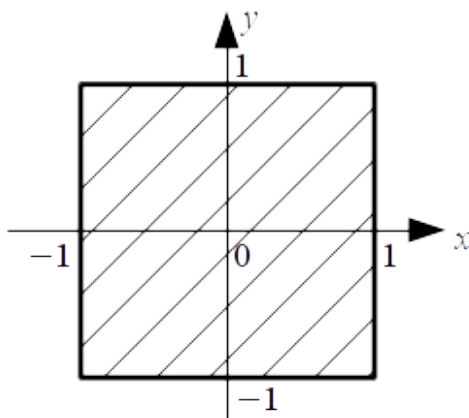


Рисунок 1.18 – Двовимірна квадратна область

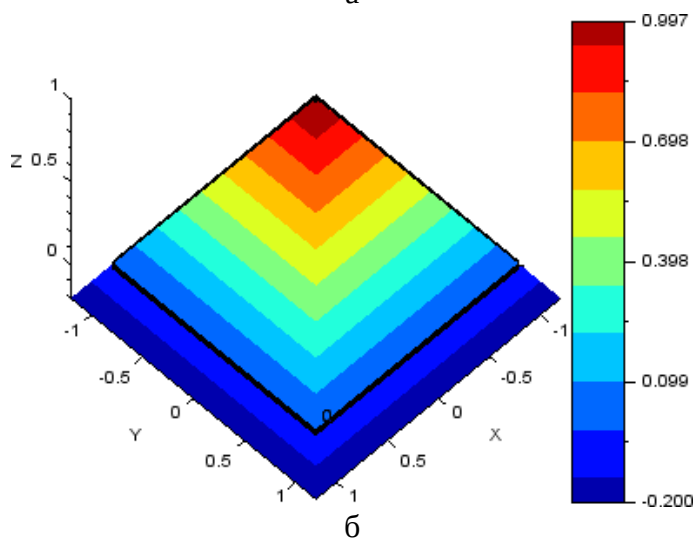
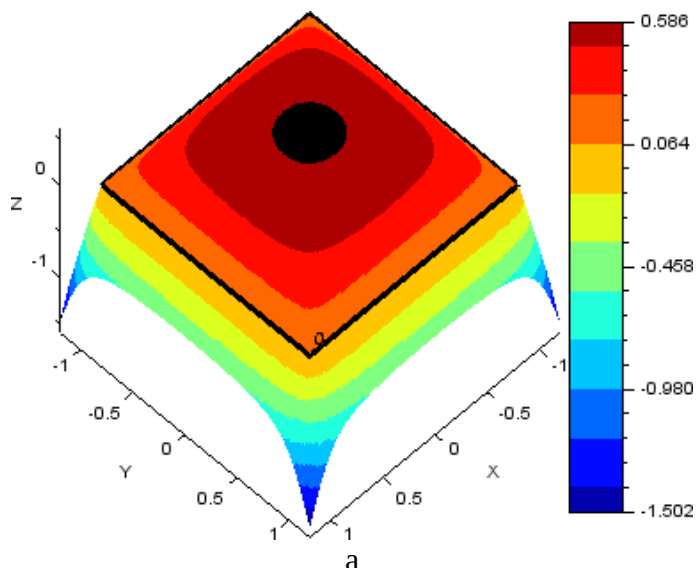


Рисунок 1.19 – Розподіл значень функцій, що подають квадратну область:

а – $F(x, y) = 2 - x^2 - y^2 - \sqrt{(1 - x^2)^2 + (1 - y^2)^2}$;

б – $Q(x, y) = \min(1 + x, 1 - x, 1 + y, 1 - y)$

Проте, за перших двох способах для опису складних областей доведеться формувати композиції описів або системи систем, що є серйозним конструктивним недоліком, що виявляється в багатьох ситуаціях, де потрібно враховувати геометричну інформацію на аналітичному рівні [271, с. 65]. Отже, опис множини єдиним рівнянням або нерівністю найбільш перспективний.

Нехай деяка дійсна функція $F(P) \geq 0$ для довільної точки P підмножини Ω евклідового простору, що відповідає геометричній формі об'єкта. В результаті, отримаємо неявне подання множини у вигляді

$$\Omega = \{P | F(P) \geq 0\}. \quad (1.5)$$

Відповідно, доповнення множини Ω – множина $\bar{\Omega}$, буде мати вигляд

$$\bar{\Omega} = \{P | -F(P) > 0\}. \quad (1.6)$$

Функцію $F(P)$ можна побудувати як суперпозицію функцій $f_i(P) \geq 0$, які відповідають примітивним формам.

Найбільш прості форми (півпростори, півплощини, циліндричні об'єкти тощо) можна подати багаточленами, елементарними функціями та константами. Проте, в інженерній практиці необхідно описувати досить складні геометричні форми об'єктів, що найбільш просто здійснювати з використанням теоретико-множинних операцій об'єднання, перетину та доповнення.

Наприклад, можна побачити, що квадрат є перетином множин, заданих нерівностями $f_1(x, y) = 1 + x \geq 0$, $f_2(x, y) = 1 - x \geq 0$, $f_3(x, y) = 1 + y \geq 0$, $f_4(x, y) = 1 - y \geq 0$. Цю область можна подати єдиною нерівністю (рис. 1.19, б):

$$Q(x, y) = \min(f_1(x, y), f_2(x, y), f_3(x, y), f_4(x, y)) \geq 0,$$

де функція знаходження мінімуму $\min$ відповідає операції перетину множин.

Використання функції $\min$ у даному випадку обумовлено спостереженням, що вона невід'ємна тоді і тільки тоді, коли її обидва аргументи невід'ємні. Тобто, якщо $\Omega_1 = \{P | f_1(P) \geq 0\}$ і $\Omega_2 = \{P | f_2(P) \geq 0\}$, тоді

$$\Omega_1 \cap \Omega_2 = \{P | \min(f_1(P), f_2(P)) \geq 0\}. \quad (1.7)$$

аналогічно функція $\max$ буде невід'ємною, якщо хоча б один її аргумент невід'ємний, що дозволяє використовувати її для подання об'єднання множин:

$$\Omega_1 \cup \Omega_2 = \{P \mid \max(f_1(P), f_2(P)) \geq 0\}. \quad (1.8)$$

Множини, які можна подати як скінченну комбінацію теоретико-множинних операцій над множинами вигляду (1.5), називають напіваналітичними [45, 55, 80]. Такі множини активно використовуються в комп'ютерній графіці й автоматизованому проектуванні.

Використання функцій $\min$ і $\max$ для опису об'єднання та перетину форм об'єктів, поданих функціонально, вперше було запропоновано в роботі [279]. Їх використання дозволяє описувати форми об'єктів шляхом логічного конструювання, але не дозволяє визначити диференціальні характеристики межі множини, що відповідає формі об'єкта.

Отже, функцію $F(P)$, що відповідає множини Ω , можна побудувати різними способами. Теорія R-функцій академіка Рвачова В. Л. [61–68, 278] пояснює, систематизує й узагальнює математичний апарат, необхідний для побудови теоретично необмеженої множини форм і відповідних їм функцій. Але залишається актуальною задача спрощення процесів побудови і верифікації функцій, що подають форми складних об'єктів.

З точки зору моделювання форм об'єктів, основне призначення теорії R-функцій – це заміна логічних або теоретико-множинних операцій відповідними їм дійсними функціями при побудові неявного подання $F(P) \geq 0$ для форми довільного об'єкта. Такі дійсні функції дійсних аргументів називають логічно змінними (їх знак повністю визначається знаками аргументів і не залежить від величини значень аргументів).

Хоча в роботах Рвачова В. Л. [61–68, 278] показана принципова можливість побудови дійсних функцій (R-відображень), відповідних до k-значної логіки, на практиці, зазвичай, досить функцій двозначної логіки. У такому випадку дійсна функція $F: \mathbb{R}^n \rightarrow \mathbb{R}$ називається R-функцією, якщо існує функція двозначної логіки $\Phi: \mathbf{B}_2^n \rightarrow \mathbf{B}_2$, яка разом з F утворює комутативну діаграму

$$\begin{array}{ccccc} \mathbb{R}^n & \xrightarrow{F} & \mathbb{R} & & \\ S_2^n \downarrow & & \downarrow & S_2, & \\ \mathbf{B}_2^n & \xrightarrow{\Phi} & \mathbf{B}_2 & & \end{array}$$

де $\mathbf{B}_2 = \{0, 1\}$;

$S_2: \mathbb{R} \rightarrow \mathbf{B}_2$ – сюр'єкція, яку можна визначити функцією Гевісайда [80]:

$$S_2(x) = \begin{cases} 0, & x \leq -0, \\ 1, & x \geq +0. \end{cases}$$

Функція двозначної логіки $\Phi: \mathbf{B}_2^n \rightarrow \mathbf{B}_2$ супроводжує R-функцію $F: \mathbb{R}^n \rightarrow \mathbb{R}$. Функцію Φ , зазвичай, визначають з використанням логічних операцій: $\wedge$ (кон'юнкція), $\vee$ (диз'юнкція) і $\neg$ (заперечення) над n логічними змінними. Ці три логічні операції утворюють повну (хоча не мінімальну) систему в множині булевих функцій.

Основні властивості R-функцій, серед яких замкнутість щодо композиції, доведені у роботі [63]. Множина R-функцій нескінченна. Проте, для додатків немає необхідності знати всі R-функції, а досить мати можливість їх конструювати.

Функція $F: \mathbb{R}^n \rightarrow \mathbb{R}$ є неявною функцією та, отже, її можна використовувати для подання форм об'єктів γ -множинами. У більшості випадків для побудови F використовують систему R-функцій

$$\begin{cases} \neg x = -x, \\ x \wedge_{\alpha} y = \frac{1}{1+\alpha} (x + y - \sqrt{x^2 + y^2 - 2\alpha xy}), \\ x \vee_{\alpha} y = \frac{1}{1+\alpha} (x + y + \sqrt{x^2 + y^2 - 2\alpha xy}), \end{cases} \quad (1.9)$$

де x і y – значення деяких неявних функцій;

$\alpha = \alpha(x, y)$ – дійсна функція, для якої справедлива нерівність $-1 \leq \alpha \leq 1$.

Найбільш часто обирають $\alpha = 0$; 1; 0,9 або $\alpha(x, y) = (1 + x^2 + y^2)^{-1}$ [70]. Розподіли значень функцій кон'юнкції та диз'юнкції у квадраті $[-1; 1] \times [-1; 1]$ за різних значеннях α наведено на рис. 1.20.

Отже, при використанні системи R-функцій (1.9) теоретико-множинні операції об'єднання (1.7) і перетину (1.8) отримують вигляд

$$\begin{aligned} \Omega_1 \cup \Omega_2 &= \{P \mid f_1(P) \vee_{\alpha} f_2(P) \geq 0\} = \\ &= \left\{ P \mid \frac{1}{1+\alpha} (f_1(P) + f_2(P) + \sqrt{f_1^2(P) + f_2^2(P) - 2\alpha f_1(P) f_2(P)}) \geq 0 \right\}, \\ \Omega_1 \cap \Omega_2 &= \{P \mid f_1(P) \wedge_{\alpha} f_2(P) \geq 0\} = \\ &= \left\{ P \mid \frac{1}{1+\alpha} (f_1(P) + f_2(P) - \sqrt{f_1^2(P) + f_2^2(P) - 2\alpha f_1(P) f_2(P)}) \geq 0 \right\}. \end{aligned}$$

Доповнення множини, визначеної функціонально, збереже вигляд (1.6). У подальшому викладенні матеріалу $\alpha = 0$, якщо інакше не зазначено.

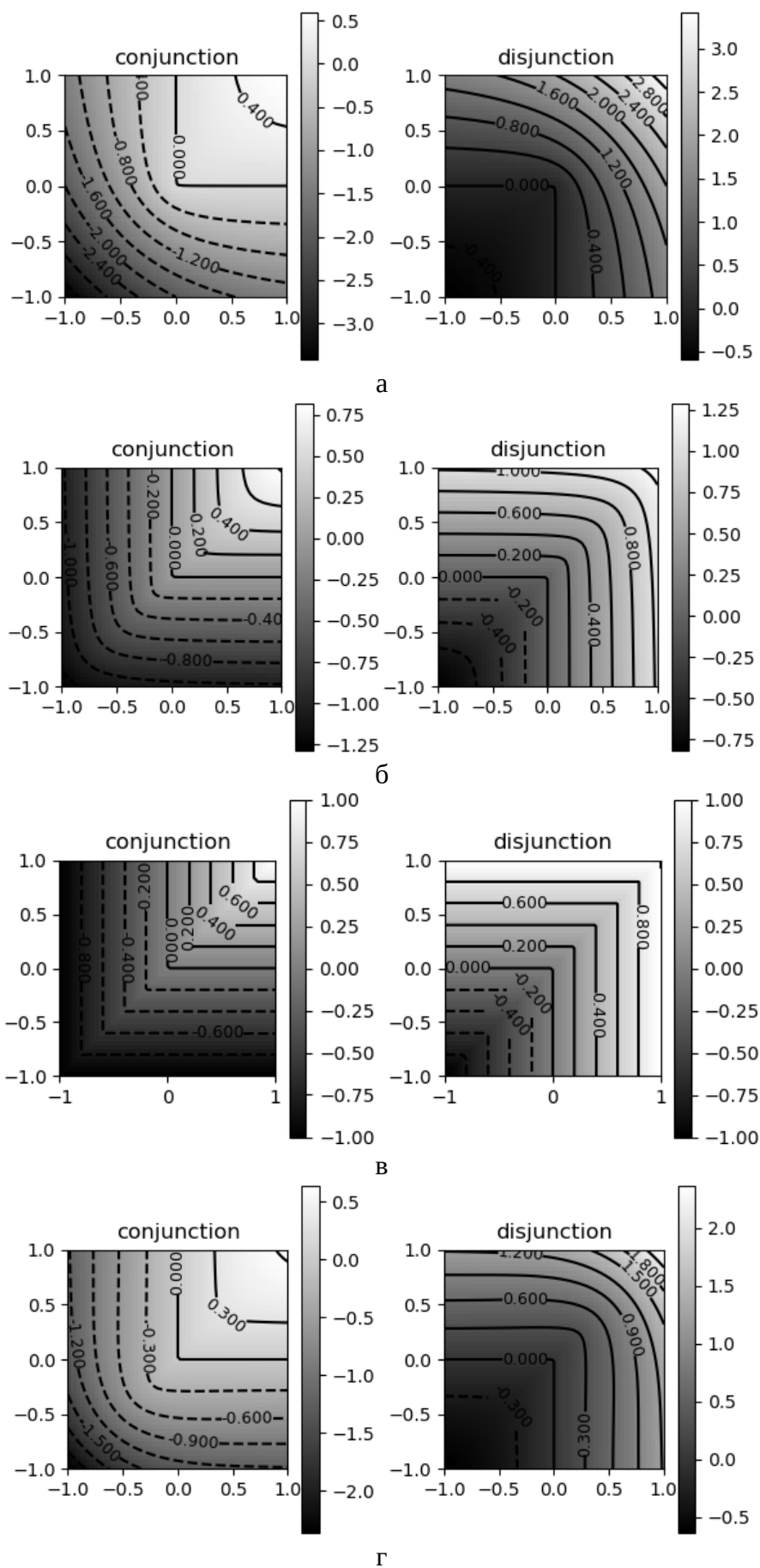


Рисунок 1.20 – Розподіли значень функцій кон'юнкції та диз'юнкції у квадраті $[-1; 1] \times [-1; 1]$: а – $\alpha = 0$; б – $\alpha = 0,9$; в – $\alpha = 1$; г – $\alpha(x, y) = (1+x^2+y^2)^{-1}$

1.6.1 Моделювання двовимірних форм об'єктів на базі неявних функцій

Двовимірні математичні моделі форм об'єктів найчастіше використовують для подання призматичних тіл зі сталою товщиною (пластин, балок тощо), в яких можна знехтувати тривимірними ефектами. У даному випадку примітивами виступають множини точок на площині вигляду (1.5), подані дійсними функціями двох дійсних змінних. Базові примітиви, зазвичай, відповідають найпростішим формам, поданим елементарними функціями. Їх суперпозиції, отримані з використанням системи (1.9), визначають нові, більш складні форми.

1.6.1.1 Примітиви

1. Півплощини

Найпростішою формою є пряма. Вона ділить площину на дві півплощини. Наприклад, множина точок, розташованих нижче прямої $y = y_0$ (включно цю пряму), буде визначена неявною функцією $ru_d(x, y) = y_0 - y$. Функція $ru_t(x, y) = \neg ru_d(x, y) = y - y_0$ визначить півплощину, розташовану вище. Аналогічно, функція $rx_l(x, y) = x_0 - x$ буде відповідати півплощині, розташованій ліворуч прямої $x = x_0$, а функція $rx_r(x, y) = \neg rx_l(x, y) = x - x_0$ визначить півплощину, розташовану праворуч. На практиці пряму зручніше визначати парою точок, через які вона проходить. Якщо $P_1(x_1; y_1)$ і $P_2(x_2; y_2)$ – дві різні точки, тоді функція

$$\text{line}(x, y, P_1, P_2) = \frac{(y - P_{1y})(P_{2x} - P_{1x}) - (x - P_{1x})(P_{2y} - P_{1y})}{\sqrt{(P_{2x} - P_{1x})^2 + (P_{2y} - P_{1y})^2}} \quad (1.10)$$

є нормальним орієнтованим рівнянням прямої [63], яке визначає півплощину, розташовану ліворуч під час руху по прямій в напрямку від точки P_1 до P_2 (рис. 1.21). У формулі (1.10) P_{ix} і P_{iy} позначають, відповідно, абсцису й ординату точки P_i , $i = \{1, 2\}$.

Розподіл значень функції $\text{line}(x, y, (-1; 1), (1; -0.5))$ у квадраті $[-2; 2] \times [-2; 2]$ наведено на рис. 1.22.

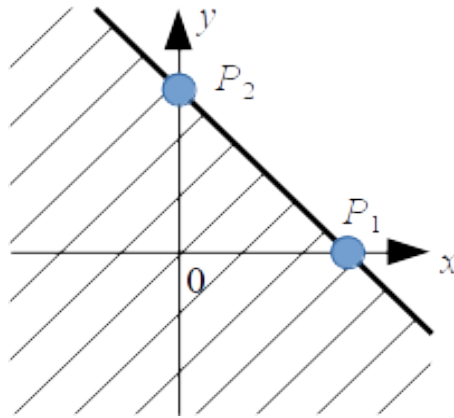


Рисунок 1.21 – Приклад півплощини, визначеної формулою (1.10)

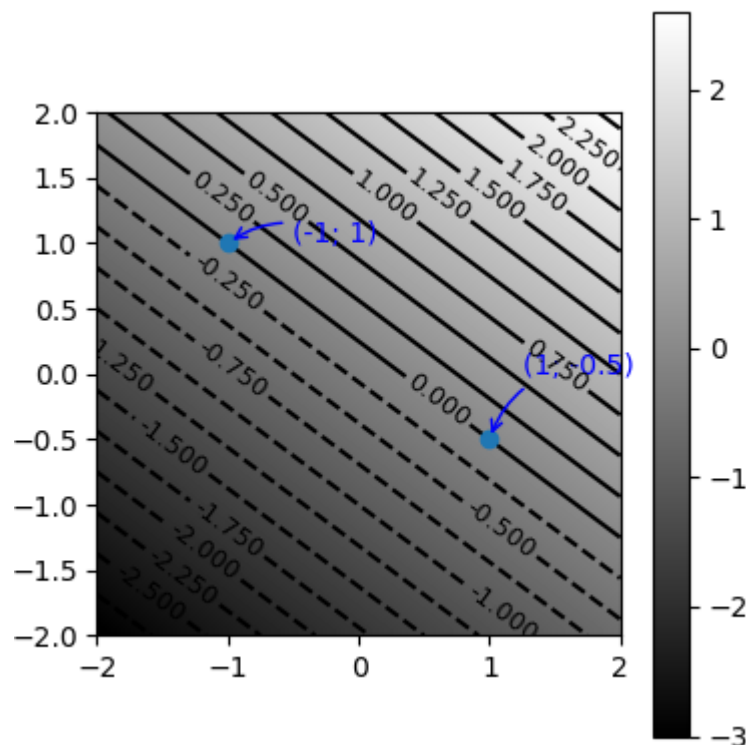


Рисунок 1.22 – Приклад розподілу значень функції (1.10)

2. Еліпси та кола

До примітивів, які визначають елементарними функціями, можна віднести області, обмежені еліпсами або, в окремому випадку – колами.

Область з центром на початку координат, обмежена еліпсом з півосями, рівними a і b , можна подати такою функцією:

$$\text{ellipse}(x, y, a, b) = 1 - \frac{x^2}{a^2} - \frac{y^2}{b^2}, \quad a > 0, \quad b > 0.$$

Якщо $a=b=r$, отримаємо область, обмежену колом радіуса r . Проте, з метою зменшення кількості арифметичних операцій, що використовується, таку область краще подати функцією

$$\text{circle}(x, y, r) = 1 - \frac{x^2 + y^2}{r^2}, \quad r > 0. \quad (1.11)$$

Приклади розподілів функцій ellipse і circle наведено на рис. 1.23.

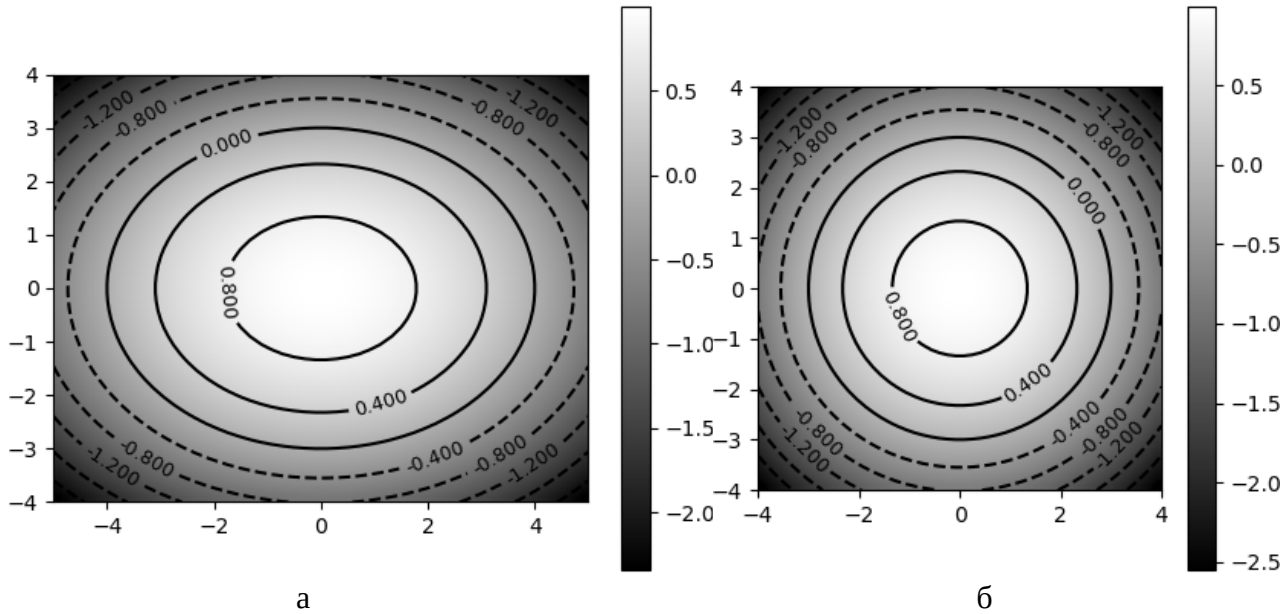


Рисунок 1.23 – Приклади розподілів значень функцій еліпсів і кіл:

а – $a=4$, $b=3$; б – $r=3$

3. Полігони

Довільний опуклий полігон (рис. 1.24, а) функціонально можна подати кон'юнкцією півплощин, визначених формулою (1.10). Наприклад, якщо його вершини впорядковані у порядку обходу проти годинникової стрілки, то опуклий полігон можна задати формулою

$$\text{convex}(x, y, P_1, P_2, \dots, P_n) = \text{line}(x, y, P_1, P_2) \wedge \text{line}(x, y, P_2, P_3) \wedge \dots \wedge \text{line}(x, y, P_{n-1}, P_n) \wedge \text{line}(x, y, P_n, P_1), \quad n \geq 3, \quad (1.12)$$

де $P_1, P_2, \dots, P_n$ – координати його вершин;

n – кількість вершин.

У формулі (1.12), а також в наступних, для економії місця в позначеннях кон'юнкції та диз'юнкції опущений символ α , що використаний у формулі (1.9). У чисельному аналізі та при візуалізації за замовчуванням прийнято $\alpha=0$ (якщо будуть використовуватися інші значення, то це буде зазначено безпосередньо у місці їх використання).

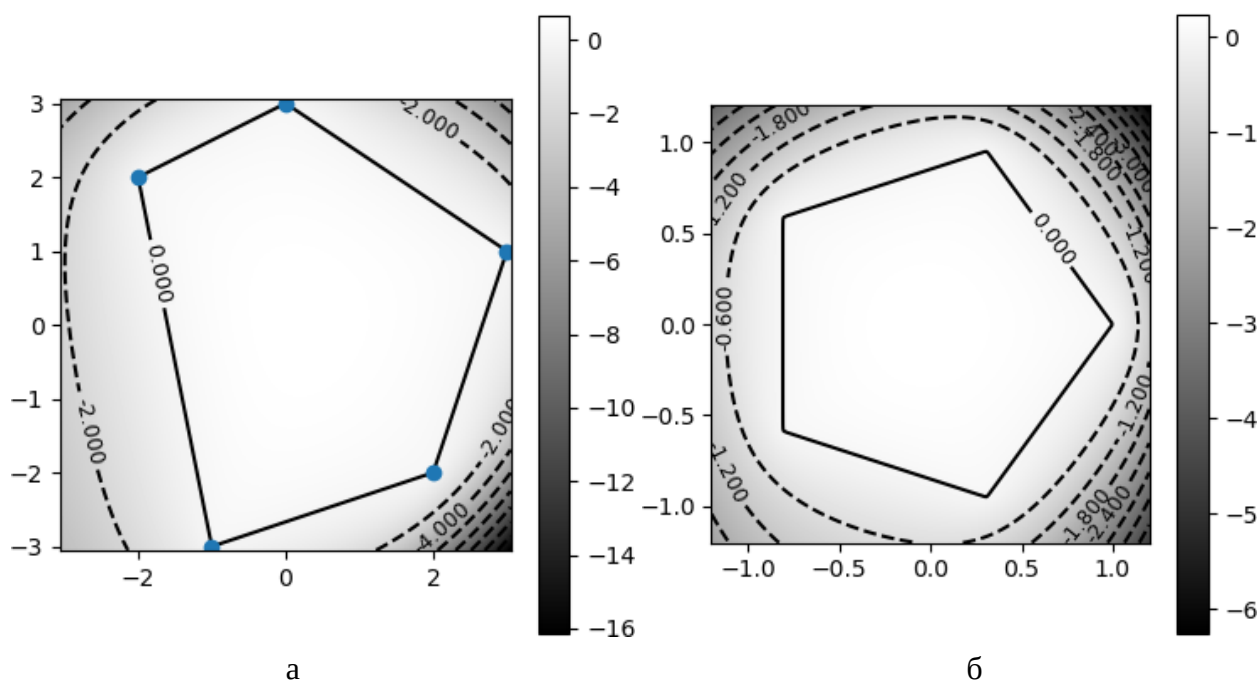


Рисунок 1.24 – Приклади розподілу значень функцій: а – convex , якщо $P_1(2;-2)$, $P_2(3;1)$, $P_3(0;3)$, $P_4(-2;2)$, $P_5(-1;-3)$; б – regular , якщо $r=1$, $n=5$

Окремим, але вельми поширеним, випадком формули (1.12) є область, обмежена правильним багатокутником (рис. 1.24, б):

$$\begin{aligned} \text{regular}(x, y, r, n) &= \text{convex}(x, y, P_1, P_2, \dots, P_n), \\ P_i &= (r \cos(\alpha_i); r \sin(\alpha_i)), \quad \alpha_i = \frac{2\pi}{n}(i-1), \quad i=1, 2, \dots, n, \quad n \geq 3. \end{aligned} \quad (1.13)$$

Такі фігури як квадрат, паралелограм, прямокутник, ромб також є окремими випадками опуклого багатокутника. Проте, з обчислювальної точки зору їх краще розглядати як самостійні форми. Наприклад, неявну функцію, що відповідає прямокутнику з шириною w і висотою h як перетин вертикальної та горизонтальної смуг:

$$\text{rectagle}(x, y, w, h) = \text{stripe}(x, w) \wedge \text{stripe}(y, h), \quad w > 0, \quad h > 0, \quad (1.14)$$

де функція

$$\text{stripe}(x, w) = 1 - \left(\frac{2}{w} x \right)^2 \quad (1.15)$$

відповідає перпендикулярній до заданої осі смузі простору ширини w .

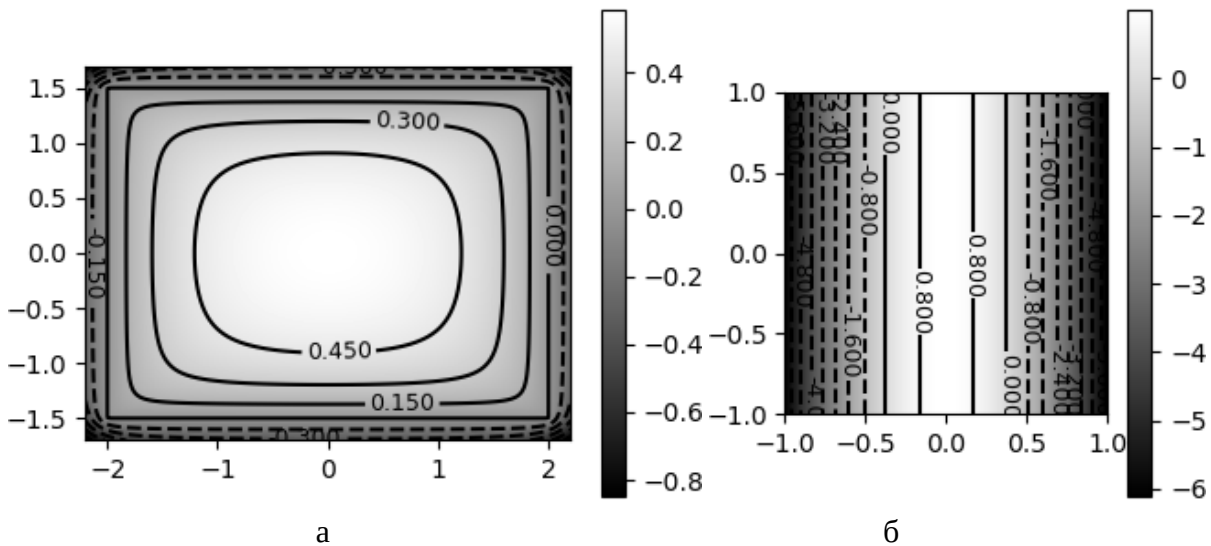


Рисунок 1.25 – Приклади розподілу значень функцій:
 а – `rectangle` , якщо $w=4$, $h=3$; б – `stripe` , якщо $w=1$

На основі формули (1.14) можна побудувати інший поширений примітив – прямокутник з округленими кутами , який може бути отриманий як об'єднання двох прямокутних областей з чотирма круглими. У такому випадку підсумкова формула набуде вигляду

$$\begin{aligned}
 \text{rectangle}_r(x, y, w, h, r) = & \text{rectangle}(x, y, w-2r, h) \vee \\
 & \vee \text{rectangle}(x, y, w, h-2r) \vee \\
 & \vee \text{circle}\left(x - \frac{w}{2} + r, y - \frac{h}{2} + r, r\right) \vee \text{circle}\left(x + \frac{w}{2} - r, y - \frac{h}{2} + r, r\right) \vee \\
 & \vee \text{circle}\left(x + \frac{w}{2} - r, y + \frac{h}{2} - r, r\right) \vee \text{circle}\left(x - \frac{w}{2} + r, y + \frac{h}{2} - r, r\right), \\
 & 0 \leq r \leq \min\left(\frac{w}{2}; \frac{h}{2}\right), \quad w > 0, \quad h > 0,
 \end{aligned} \tag{1.16}$$

де функція `rectangle` визначена формулою (1.14);

функція `circle` має вигляд (1.11);

w – ширина прямокутника з округленими кутами;

h – його висота;

r – радіус округлення.

4. Рух та обертання двовимірних об'єктів

Використання неявних функцій дозволяє відносно просто маніпулювати формами за допомогою операцій відображення, переміщення та обертання. Такі операції також можна подати як неявні функції.

Зрозуміло, якщо функція $F(x, y)$ визначає геометричну форму деякого об'єкта Ω , то функція $F(-x, y)$ визначає симетричну щодо осі Oy форму, $F(x, -y)$ – відносно осі Ox , а $F(-x, -y)$ – відносно початку координат. Якщо необхідно визначити форму об'єкта, всі точки якого зміщені на вектор $\mathbf{v}=[x_0; y_0]^T$ відносно Ω , то можна використовувати функцію $F(x-x_0, y-y_0)$. У випадках, коли необхідно повернути форму об'єкта на кут α , використовують обертання системи координат: $F(x \cos(\alpha) + y \sin(\alpha), -x \sin(\alpha) + y \cos(\alpha))$.

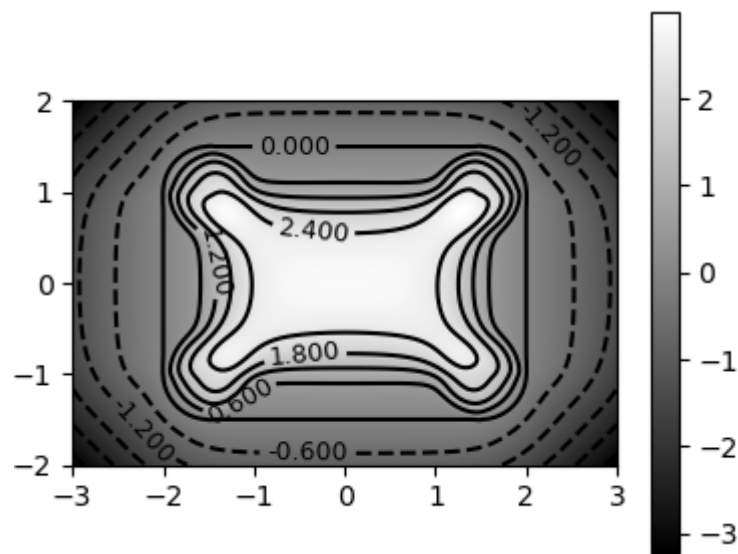


Рисунок 1.26 – Приклад розподілу значень функції rectangle_r : $w=4$, $h=3$ і $r=0,5$

Аналіз більш складних перетворень координат і суперпозицій з періодичними функціями для функціонального подання форм об'єктів з різними типами симетрій виконаний в роботах [69–76].

1.6.1.2 Приклади функціонального подання креслень двовимірних форм об'єктів

1. L-область.

Розглянемо L-область, зображену на рис. 1.27. Цю область можна подати об'єднанням двох областей, обмежених прямокутниками. У результаті отримаємо таку функцію:

$$\begin{aligned} \text{lshape}(x, y, w, h, a) = \\ = \text{rectangle}\left(x - \frac{w}{2}, y - \frac{a}{2}, w, a\right) \vee \text{rectangle}\left(x - \frac{a}{2}, y - \frac{h}{2}, a, h\right), \end{aligned} \quad (1.17)$$

де rectangle – функція вигляду (1.14).

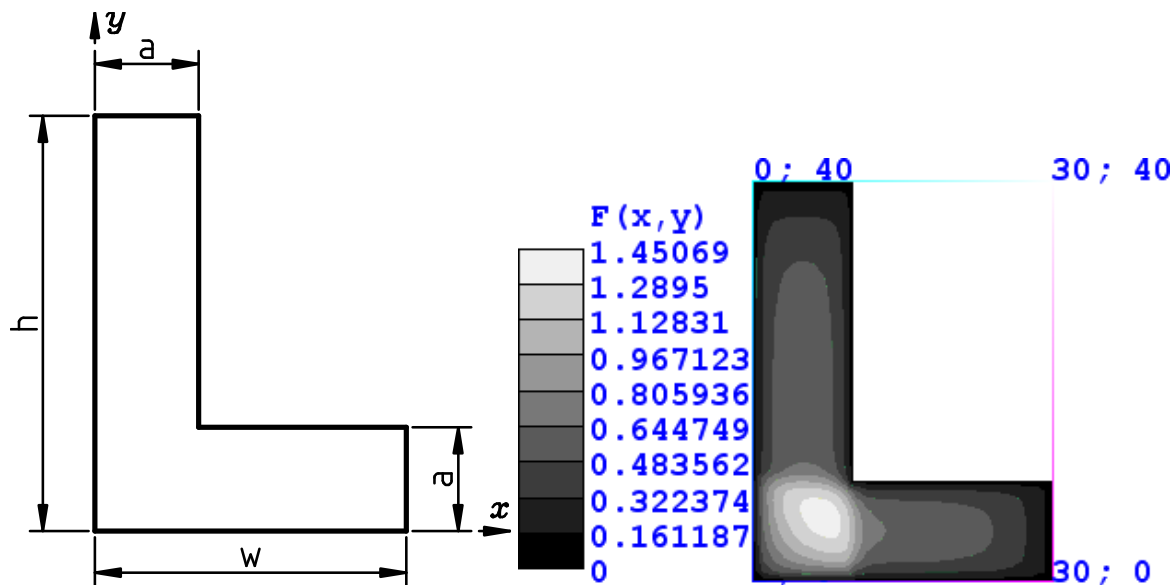


Рисунок 1.27 – L-область: креслення та розподіл значень функції (1.17), якщо $w=30$, $h=40$ і $a=10$

2. Квадратна пластина з трикутним отвором у центрі.

Квадратну пластину, що має трикутний отвір у центрі, (рис. 1.28) можна подати з використанням функції (1.13) у такому вигляді:

$$\text{qplate}(x, y, a, r) = \text{rectangle}(x, y, a, a) \wedge \neg \text{regular}(x, y, r, 3). \quad (1.18)$$

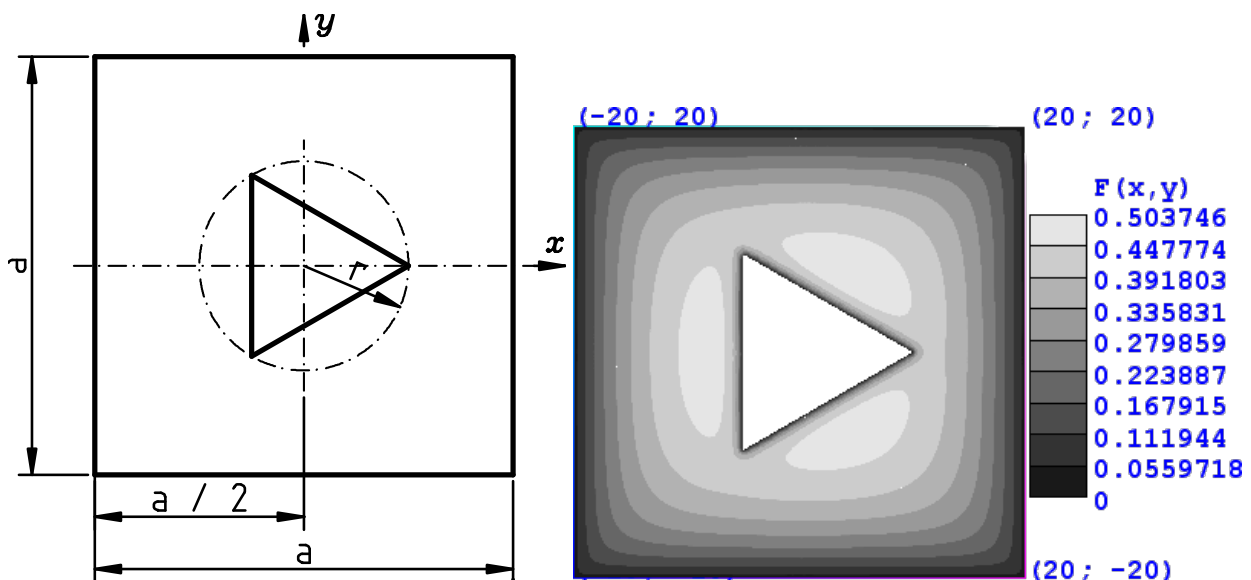


Рисунок 1.28 – Квадратна пластина з трикутним отвором: креслення і розподіл значень функції (1.18), якщо $a=40$ і $r=10$

3. Профіль гайки.

Двовимірну область у формі гайки (рис. 1.29), також можна задати за допомогою формул (1.11) і (1.13) (як логічну різницю області, обмеженої правильним шестикутником, і області, обмеженої колом):

$$\text{nut}(x, y, r_1, r_2) = \text{regular}(x, y, r_1, 6) \wedge \neg \text{circle}(x, y, r_2), \quad (1.19)$$

де радіуси r_1 і r_2 , що зображені на рис. 1.29.

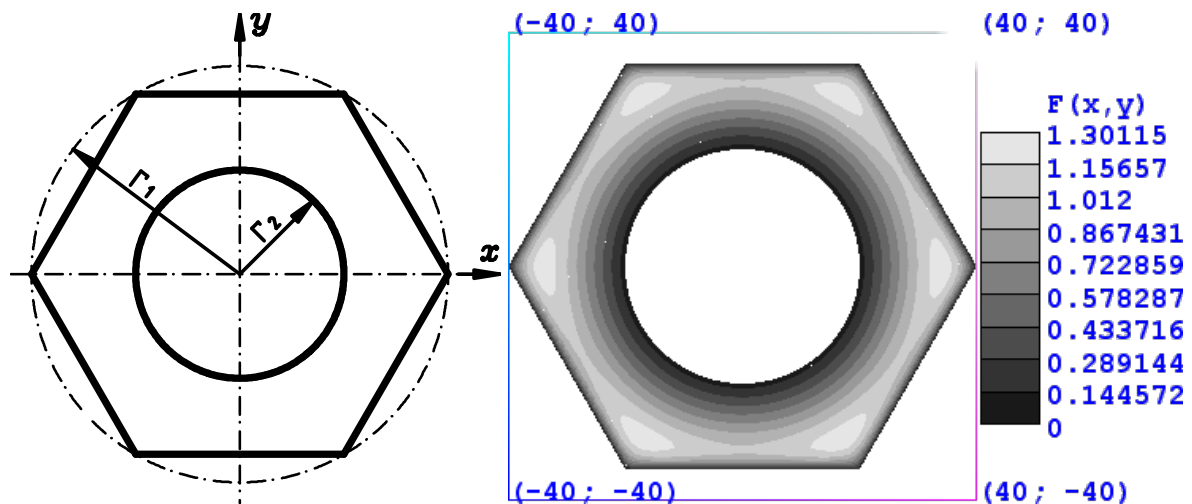


Рисунок 1.29 – Модель двовимірної гайки: креслення і розподіл значень функції (1.19), якщо $r_1=40$ і $r_2=20$

4. Фрикційний диск трансмісії.

Фрикційний диск трансмісії можна подати як геометричний об'єкт з точковою симетрією циклічного типу у вигляді такої функції:

$$\begin{aligned} \text{clutch}(x, y, r, R, S, w, n) = \\ = \left[\text{stripe} \left(\sin \left(\mu(n, \arctg_2(y, x)) \sqrt{x^2 + y^2} \right), w \right) \wedge \text{circle}(x, y, S) \vee \text{circle}(x, y, R) \right] \wedge \\ \wedge \neg \text{circle}(x, y, r), \end{aligned} \quad (1.20)$$

де r, R, S і w – розміри, зображені на рис. 1.30;

n – кількість зубців;

stripe – функція (1.15);

circle – функція (1.11);

$\mu(n, \theta)$ – часткова сума ряду Фур'є, визначена формулою [73]

$$\mu(n, \theta) = \frac{8}{\pi n} \sum_{k=1}^4 (-1)^{k+1} \sin \left[\frac{(2k-1) \frac{n\theta}{2}}{(2k-1)^2} \right]; \quad (1.21)$$

$\arctg_2(y, x)$ – спеціальна функція – арктангенс відношення y до x [280].

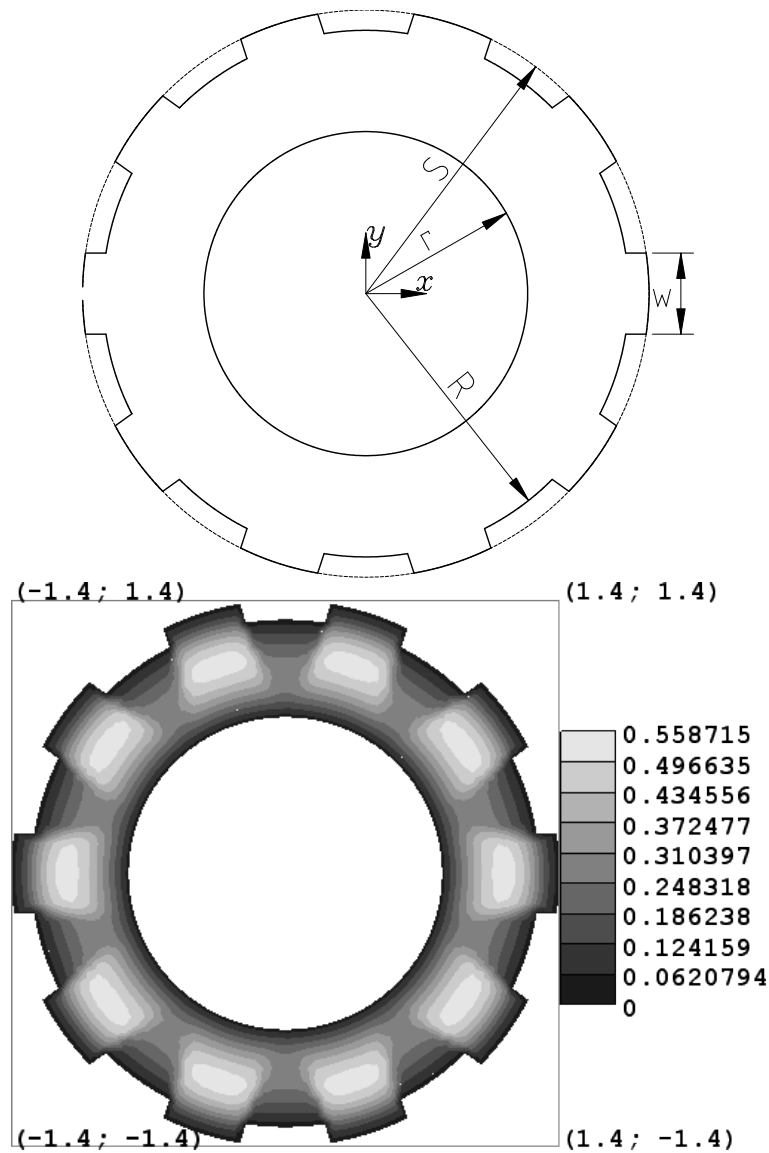


Рисунок 1.30 – Фрикційний диск трансмісії: креслення і розподіл значень функції (1.20), якщо $S=1,4$, $R=1,3$, $r=0,8$, $w=0,4$ і $n=10$

5. Планка

Контур спеціальної планки (рис. 1.31), можна подати такою формулою:

$$\begin{aligned} \text{planch}(x, y, w, h, R, r) = & \left[\text{stripe}(x, w) \wedge \text{stripe}\left(y, h + (2R - h)(2x + w) \frac{1}{2w}\right) \right] \vee \text{circle}\left(x - \frac{w}{2}, y, R\right) \Bigg\} \wedge \\ & \wedge \neg \text{circle}\left(x - \frac{w}{2}, y, r\right), \end{aligned} \quad (1.22)$$

де w – ширина планки;

h – висота планки;

R – радіус заокруглення;

r – радіус отвору.

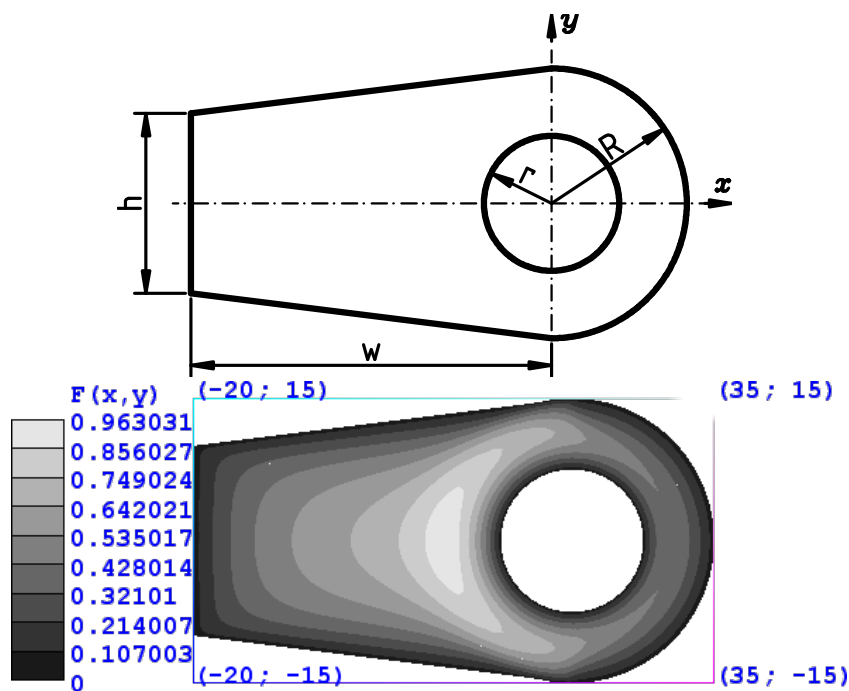


Рисунок 1.31 – Контур спеціальної планки: креслення і розподіл значень функції (1.22), якщо $h=20$, $w=40$, $R=15$ і $r=7,5$

6. Прокладка

Модель прокладки (рис. 1.32) можна подати у такому вигляді:

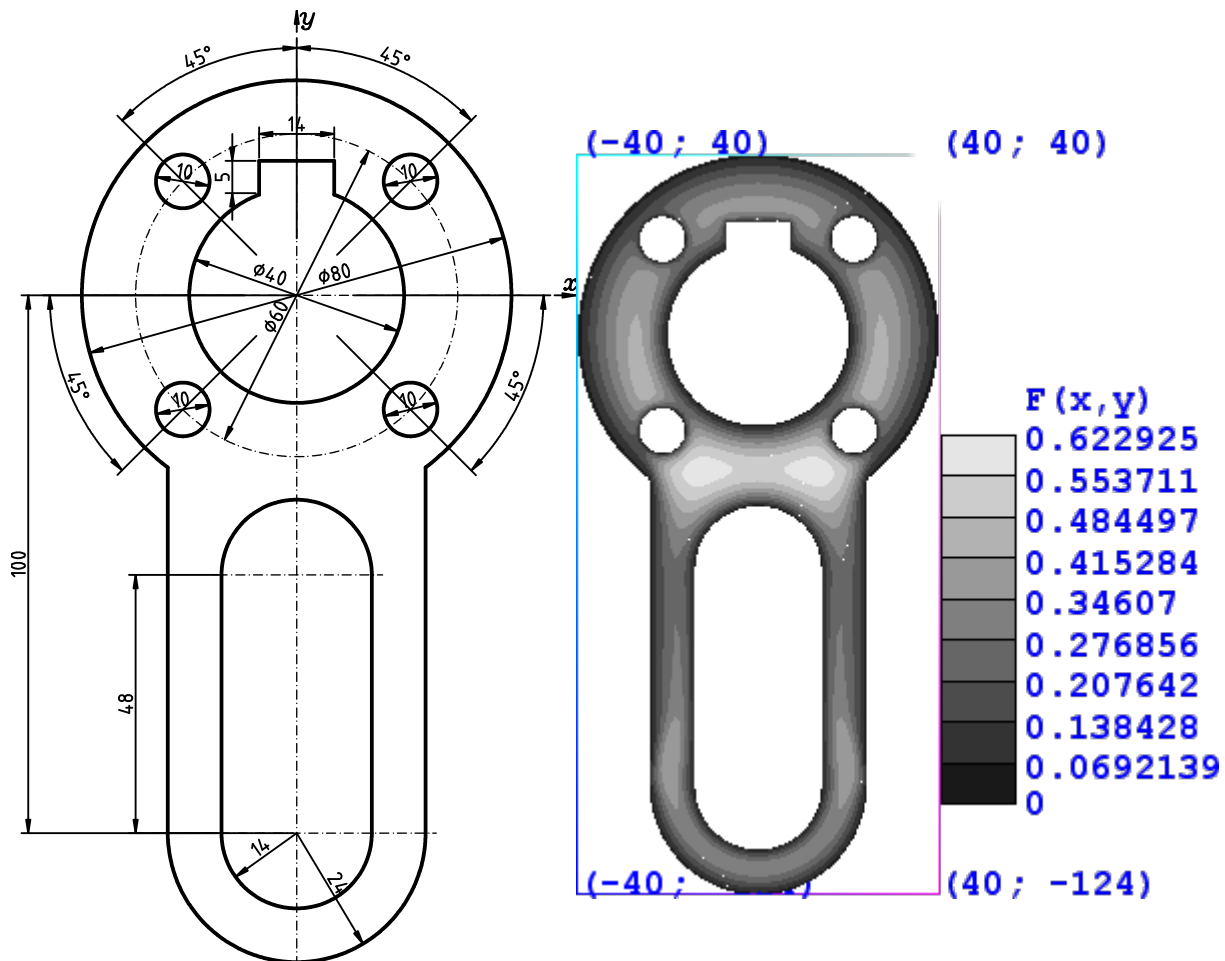
$$\begin{aligned} \text{layer}(x, y) = & \left[\text{circle}(x, y, r_1) \vee \text{circle}(x, y+h_2, r_5) \vee \text{rectangle}\left(x, y+\frac{h_2}{2}, w_2, h_2\right) \right] \wedge \\ & \wedge \neg \text{circle}(x, y, r_2) \wedge \neg \text{rectangle}(x, y-r_2, w_1, 2h_1) \wedge \\ & \wedge \neg \text{circle}\left(r_{xy} \cos\left(\mu\left(4, \theta + \frac{\pi}{4}\right)\right) - r_3, r_{xy} \sin\left(\mu\left(4, \theta + \frac{\pi}{4}\right)\right), r_4\right) \wedge \\ & \neg \text{rectangle}\left(x, y+h_2 - \frac{h_3}{2}, 2r_6, h_3\right) \wedge \neg \text{circle}(x, y+h_2, r_6) \wedge \neg \text{circle}(x, y+h_2-h_3, r_6), \end{aligned}$$

де параметри $r_1 = \frac{80}{2} = 40$, $r_2 = \frac{40}{2} = 20$, $r_3 = \frac{60}{2} = 30$, $r_4 = \frac{10}{2} = 5$, $r_5 = 24$, $r_6 = 14$, $w_1 = 14$, $h_1 = 5$, $w_2 = 2r_5$, $h_2 = 100$, $h_3 = 48$ відповідають розмірам на кресленні;

$$r_{xy} = \sqrt{x^2 + y^2};$$

$\arctg_2(y, x)$ – спеціальна функція – арктангенс відношення y до x [280];

$\mu(n, \theta)$ – функція (1.21), яка забезпечує трансляцію отвору.

Рисунок 1.32 – Прокладка: креслення і розподіл значень функції $\text{layer}(x, y)$

1.6.2 Моделювання тривимірних форм об'єктів на базі неявних функцій

Чимало моделей тривимірних примітивів можна отримати за аналогією з відповідними двовимірними. Крім того, наведені вище неявні функції у тривимірному просторі визначають нескінченні (уздовж осі Oz) призматичні області, що можна використовувати як самостійні тривимірні примітиви (наприклад, для опису форм отворів).

1.6.2.1 Примітиви

1. Півпростори

За аналогією з двовимірним випадком найпростішою формою буде площина. Вона ділить простір на два півпростори. Так, наприклад, неявна

функція $px(x, y, z) = x_0 - x$ буде відповідати півпростору, розташованому лівіше площини $x = x_0$, функція $py(x, y, z) = y_0 - y$ визначає множину точок, розташованих нижче площини $y = y_0$, а функція $pz(x, y, z) = z_0 - z$ визначить півпростір, розташований глибше площини $z = z_0$.

На практиці площину досить часто доводиться визначати трьома точками, через які вона проходить. Якщо $P_1(x_1; y_1; z_1)$, $P_2(x_2; y_2; z_2)$ і $P_3(x_3; y_3; z_3)$ – різні точки, що не лежать на одній прямій, то функція

$$\text{plane}(x, y, z, P_1, P_2, P_3) = \frac{(x - x_1)A + (y - y_1)B + (z - z_1)C}{\sqrt{A^2 + B^2 + C^2}}, \quad (1.23)$$

де x_i , y_i і z_i позначають, відповідно, абсцису, ординату і аплікату точки P_i , $i = \{1, 2, 3\}$;

$$A = (y_2 - y_1)(z_3 - z_1) - (z_2 - z_1)(y_3 - y_1);$$

$$B = (z_2 - z_1)(x_3 - x_1) - (x_2 - x_1)(z_3 - z_1);$$

$$C = (x_2 - x_1)(y_3 - y_1) - (y_2 - y_1)(x_3 - x_1).$$

За аналогією з формулою (1.10), функцію (1.23) можна вважати нормальним орієнтованим рівнянням площини.

2. Еліпсоїди, сфери та циліндри

До тривимірним примітивів, що визначаються елементарними функціями, можна віднести області, які обмежені еліпсоїдами або, в окремому випадку, сферами. Тривимірне тіло з центром на початку координат, яке обмежене еліпсоїдом з півосями, рівними a , b і c , можна подати такою функцією:

$$\text{ellipsoid}(x, y, z, a, b, c) = 1 - \frac{x^2}{a^2} - \frac{y^2}{b^2} - \frac{z^2}{c^2}, \quad a > 0, \quad b > 0, \quad c > 0.$$

Аналогічно двовимірному випадку, якщо $a = b = c = r$ отримаємо модель тіла, яке обмежене сферою радіуса r . Але з практичних міркувань таку область краще подати функцією

$$\text{sphere}(x, y, z, r) = 1 - \frac{x^2 + y^2 + z^2}{r^2}. \quad (1.24)$$

Циліндри можна подати функцією (1.11) з обмеженням по довжині твірної. Наприклад, якщо припустити, що твірна паралельна осі Oz , то такий циліндр задається формулою

$$\text{cylinder}(x, y, z, r, h) = \text{circle}(x, y, r) \wedge \text{stripe}(z, h), \quad (1.25)$$

де r – радіус основи;

h – довжина твірної;

circle – функція (1.11);

stripe – функція (1.15).

Циліндри з твірними, які спрямовані вздовж інших осей, можна отримати перестановками координат у формулі (1.25).

3. Багатогранники

Довільний опуклий багатогранник можна подати перетином (кон'юнкцією) граней, які обмежують його, у такому вигляді:

$$\begin{aligned} \text{polyhedron}(x, y, z, (P_{11}, P_{12}, P_{13}), \dots, (P_{k1}, P_{k2}, P_{k3})) = \\ = \text{plane}(x, y, z, P_{11}, P_{12}, P_{13}) \wedge \dots \wedge \text{plane}(x, y, z, P_{k1}, P_{k2}, P_{k3}), \end{aligned} \quad (1.26)$$

де k – кількість граней;

plane – функція (1.23);

P_{ij} ($i=1, \overline{k}, j=\{1, 2, 3\}$) – точки, які належать i -й грані у порядку їх обходу.

Наприклад, октаедр з ребром $\sqrt{2}$ (рис. 1.33) з використанням формули (1.26) подається у такому вигляді:

$$\begin{aligned} \text{octahedron}_2(x, y, z) = \\ = \text{polyhedron}(x, y, z, \\ (P_3, P_2, P_1), (P_1, P_2, P_4), (P_5, P_3, P_1), (P_1, P_4, P_5), \\ (P_6, P_2, P_3), (P_4, P_2, P_6), (P_6, P_3, P_5), (P_5, P_4, P_6)), \end{aligned} \quad (1.27)$$

де $P_1(0; 0; 1)$, $P_2(0; -1; 0)$, $P_3(1; 0; 0)$, $P_4(-1; 0; 0)$, $P_5(0; 1; 0)$, $P_6(0; 0; -1)$ – вершини октаедра;

polyhedron – функція (1.26).

Тривимірним аналогом формули (1.14) для прямокутника є формула (1.28), яка також заснована на функції (1.15). Формулу (1.28) можна використовувати для визначення прямокутного паралелепіпеда зі сторонами, рівними w , h і d :

$$\begin{aligned} \text{cuboid}(x, y, z, w, h, d) = \text{stripe}(x, w) \wedge \text{stripe}(y, h) \wedge \text{stripe}(z, d), \\ w > 0, h > 0, d > 0. \end{aligned} \quad (1.28)$$

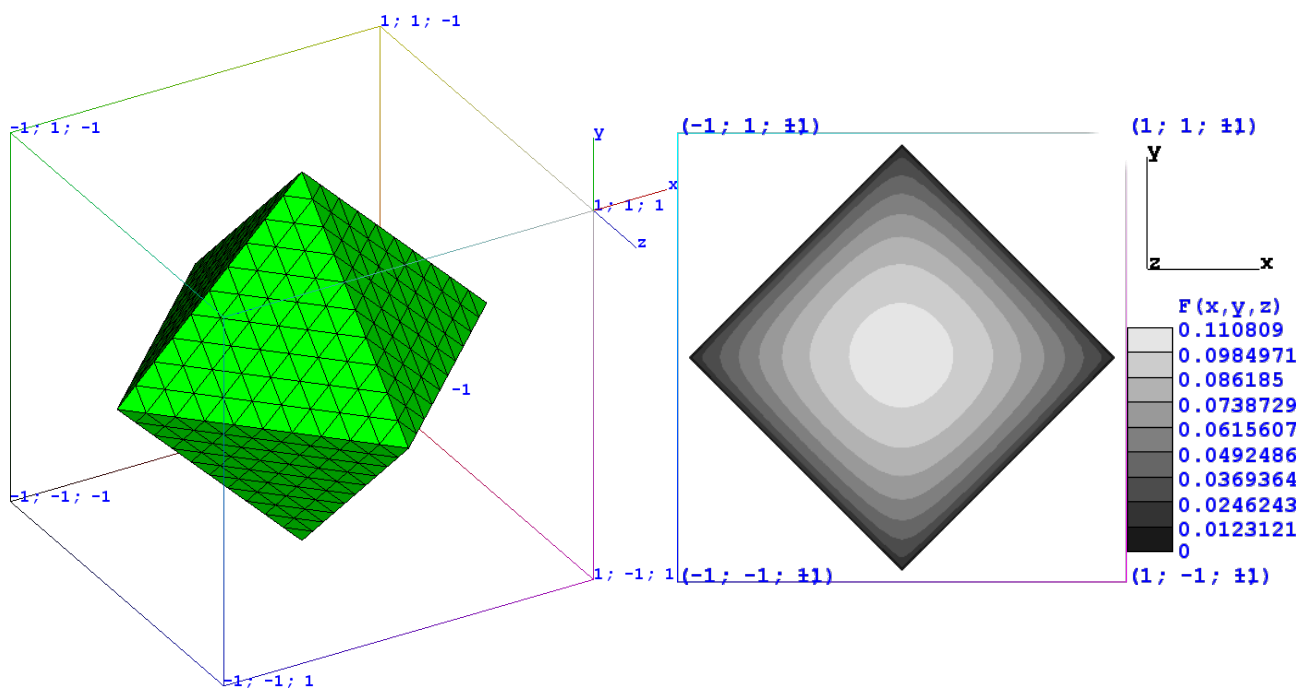


Рисунок 1.33 – Октаедр: загальний вигляд і розподіл значень функції (1.27) у перерізі площиною $z=0$

1.6.2.2 Приклади функціонального подання креслень тривимірних форм об'єктів

1. Гайка з метричною різьбою.

Гайка є циліндричним тілом (рис. 1.34). Її основа – правильний шестикутник, вписаний в коло радіуса $R = S/\sqrt{3}$, (S – розмір «під ключ»). Висота циліндра дорівнює H . У ньому є осьовий отвір з радіусом $r_i = 0,85r$ ($r = d/2$ – зовнішній радіус різьби відповідного болта, r_i – внутрішній радіус різьби), а також дві фаски: під кутом 30° на відстані $d_w/2$ від центру та під кутом 120° на відстані r_i від центру.

Якщо систему координат вибрати таким чином, щоб вісь Ox проходила через будь-яких два протилежних кута гайки, ортогональна їй вісь Oy – через середини двох протилежних граней, а вісь Oz збігалася з віссю гайки (рис. 1.34, б), то гайку можна подати такою формулою:

$$\begin{aligned} \text{nut}_{3D}(x, y, z) = \\ = \text{nut}(x, y, R, r_i) \wedge \left(\frac{H^2}{4} - z^2 \right) \wedge \neg \text{rbevel}(\sqrt{x^2 + y^2}, z) \wedge \neg \text{thread}(x, y, z), \end{aligned} \quad (1.29)$$

де функція $\text{rbevel}(x, y)$ визначає фаски в тілі обертання, її вигляд такий:

$$\begin{aligned}
 \text{rbevel}(x, y) = & \text{line}\left(x, y, \left(\frac{d_w}{2}; -\frac{H}{2}\right), \left(R; -\frac{H}{2} + b\right)\right) \wedge \\
 & \wedge \text{line}\left(x, y, \left(R; \frac{H}{2} - b\right), \left(\frac{d_w}{2}; \frac{H}{2}\right)\right) \wedge \\
 & \wedge \text{line}\left(x, y, \left(r - e; -\frac{H}{2} + c\right), \left(r; -\frac{H}{2}\right)\right) \wedge \text{line}\left(x, y, \left(r; \frac{H}{2}\right), \left(r - e; \frac{H}{2} - c\right)\right).
 \end{aligned} \quad (1.30)$$

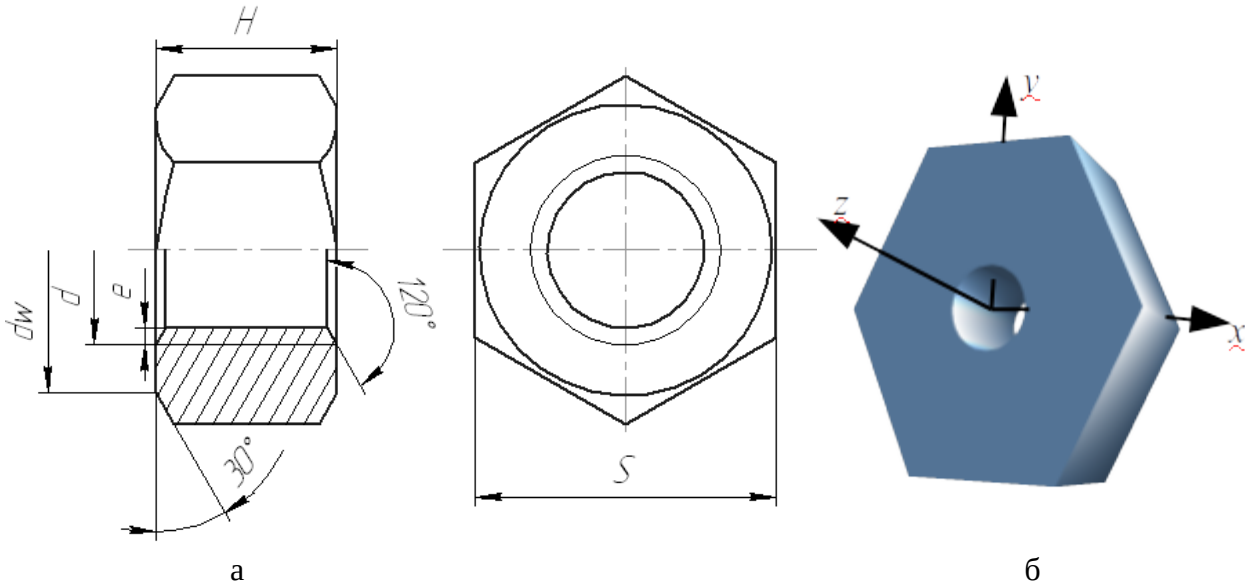


Рисунок 1.34 – Гайка: а – креслення; б – схема розташування осей координат

Функція $\text{thread}(x, y, z)$ відповідає метричній різьбі, яку можна подати як гвинтовий геометричний об'єкт, утворений рухом правильного трикутного профілю зі стороною, що дорівнює кроку різьби:

$$\begin{aligned}
 \text{thread}(x, y, z) = & \text{regular}\left(\sqrt{x^2 + y^2} - r_i + \delta, \mu(x, y, z), \frac{m}{\sqrt{3}}, 3\right) \wedge \\
 & \wedge \neg \text{circle}(x, y, r),
 \end{aligned} \quad (1.31)$$

де m – крок різьби;

$\delta = m \frac{\sqrt{3}}{12} - \frac{r - r_i}{2}$ – зсув трикутника, що забезпечує центральне положення поверхні змійовика щодо внутрішньої та зовнішньої поверхонь різьби (рис. 1.35);

$$\mu(x, y, z) = m \frac{4}{\pi^2} \sum_{k=1}^{\infty} (-1)^{k+1} \frac{\sin\left[(2k-1)\frac{\pi}{m}\left(z - \frac{m}{2\pi} \text{atan}_2(y, x)\right)\right]}{(2k-1)^2} \quad \text{– перетворення,}$$

що забезпечує повторення кілець змійовика [73];

circle – функція (1.11).

У формулах (1.29)–(1.30) введено такі позначення: $b = \left(R - \frac{d_w}{2}\right) \operatorname{tg} \frac{\pi}{6}$; $c = e \operatorname{tg} \frac{\pi}{3}$; $e = r - r_i = 0,15r$ – розмір зубця різьби; $\operatorname{nut}(x, y, r_1, r_2)$ – функція, що має вигляд (1.19); $\operatorname{line}(x, y, P_1, P_2)$ – функція (1.10). При обчисленні функції (1.31) зазвичай досить підсумовування чотирьох членів ряду. На рис. 1.36 наведено загальний вигляд поверхні гайки, визначеної формулою (1.29), і розподіл значень функції в перерізі площиною $y=0$.

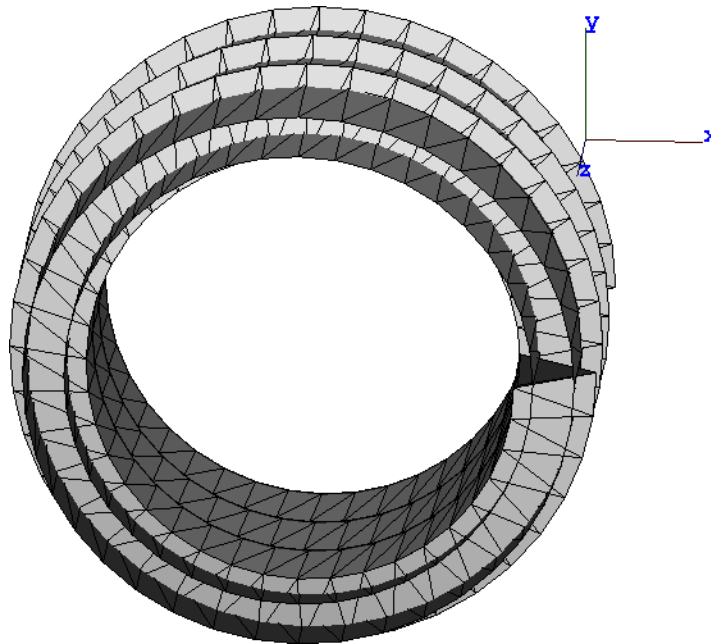


Рисунок 1.35 – Взаємне розташування поверхонь різьби

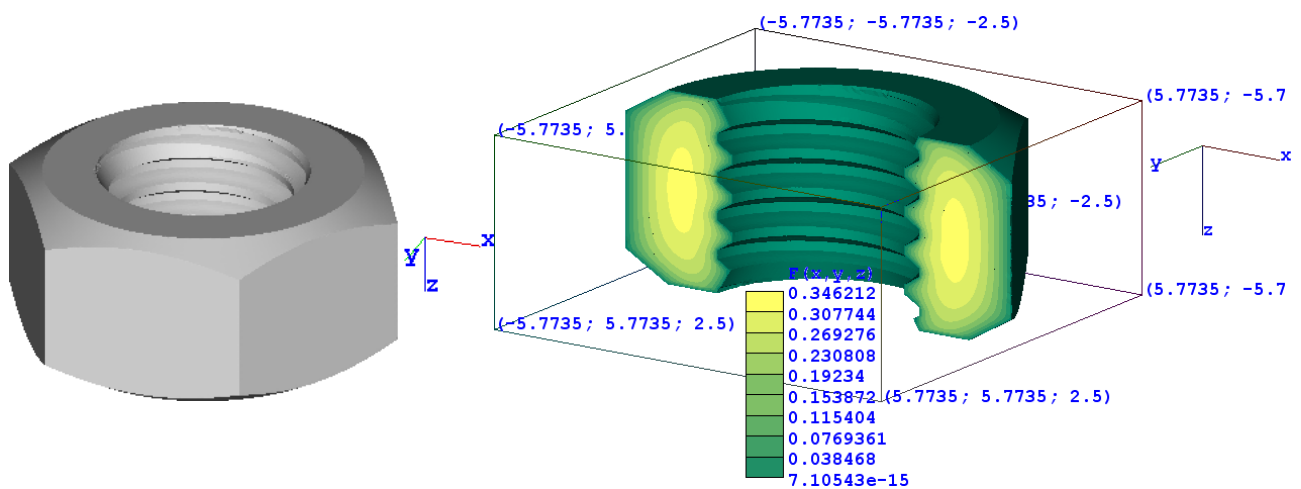


Рисунок 1.36 – Гайка: загальний вигляд і розподіл значень функції (1.29) в перерізі площиною $y=0$, якщо $S=10$, $H=5$, $d=6$ і $m=1$

2. Перехідник задньої ступиці.

Перехідник задньої ступиці (рис. 1.37) можна розглядати як циліндричне тіло з направляючою в формі прямокутника з округленими кутами. Всі кругові

вирізи, крім найбільшого, можна подати як логічне віднімання кругових областей відповідного радіуса.

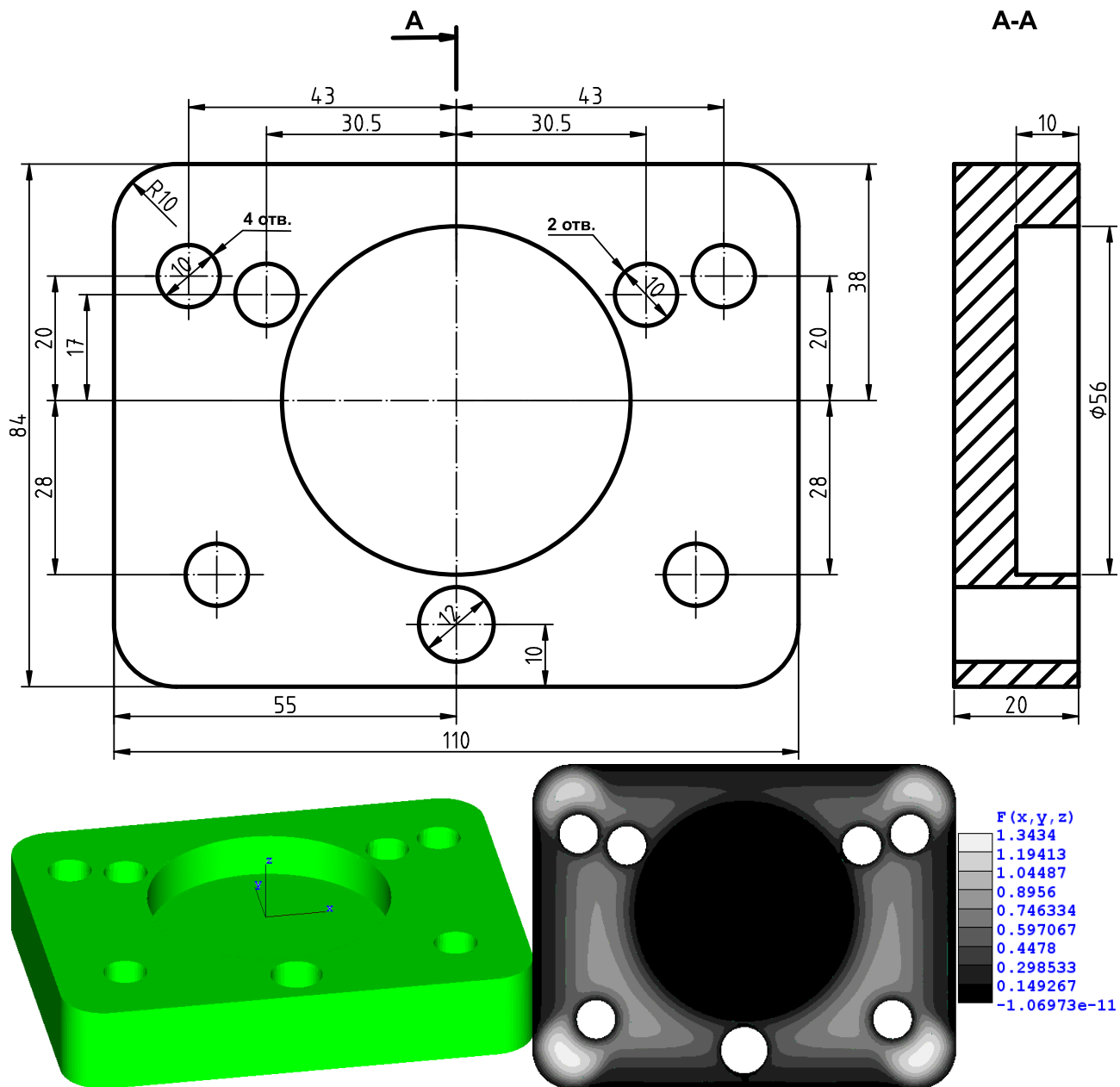


Рисунок 1.37 – Перехідник задньої ступиці: креслення, загальний вигляд і розподіл значень функції в перерізі площиною $z=0$

Якщо припустити, що площа xOy збігається з серединною площиною профілю, водночас початок координат розташовано у центрі найбільшого отвору (рис. 1.37), то перехідник задньої ступиці можна подати такою формулою:

$$\begin{aligned}
 w(x, y, z) = & \text{rectangle}_r(x, y, 110, 84, 10) \wedge \\
 & \wedge \neg \text{circle}(x-43, y-20, 5) \wedge \neg \text{circle}(x+43, y-20, 5) \wedge \\
 & \wedge \neg \text{circle}(x-38.5, y+28, 5) \wedge \neg \text{circle}(x+38.5, y+28, 5) \wedge \\
 & \wedge \neg \text{circle}(x-30.5, y-17, 5) \wedge \neg \text{circle}(x+30.5, y-17, 5) \wedge \\
 & \wedge \neg \text{circle}(x, y+36, 6) \wedge \neg [\text{circle}(x, y, 28) \wedge z] \wedge \text{stripe}(z, 20),
 \end{aligned}$$

де rectangle_r – функція (1.16);

circle – функція (1.11);

stripe – функція (1.15).

3. Подовжувач важеля передньої підвіски.

Направляючу подовжувача важеля передньої підвіски можна подати логічним об'єднанням прямокутної області з двома круговими, а також відніманням двох напівплощин, що забезпечують звуження у верхній частині (рис. 1.38), і двох кругових областей (циліндричні отвори):

$$\begin{aligned}
 p(x, y) = & [\text{circle}(x, y, 21) \vee \text{rectangle}(x, y-115/2, 42, 115) \vee \text{circle}(x, y-115, 14)] \wedge \\
 & \wedge \text{line}(x, y, (-14; 115), (-21; 81)) \wedge \text{line}(x, y, (21; 81), (14; 115)) \wedge \\
 & \wedge \neg \text{circle}(x, y-115, 5) \wedge \neg \text{circle}(x, y-72, 5).
 \end{aligned}$$

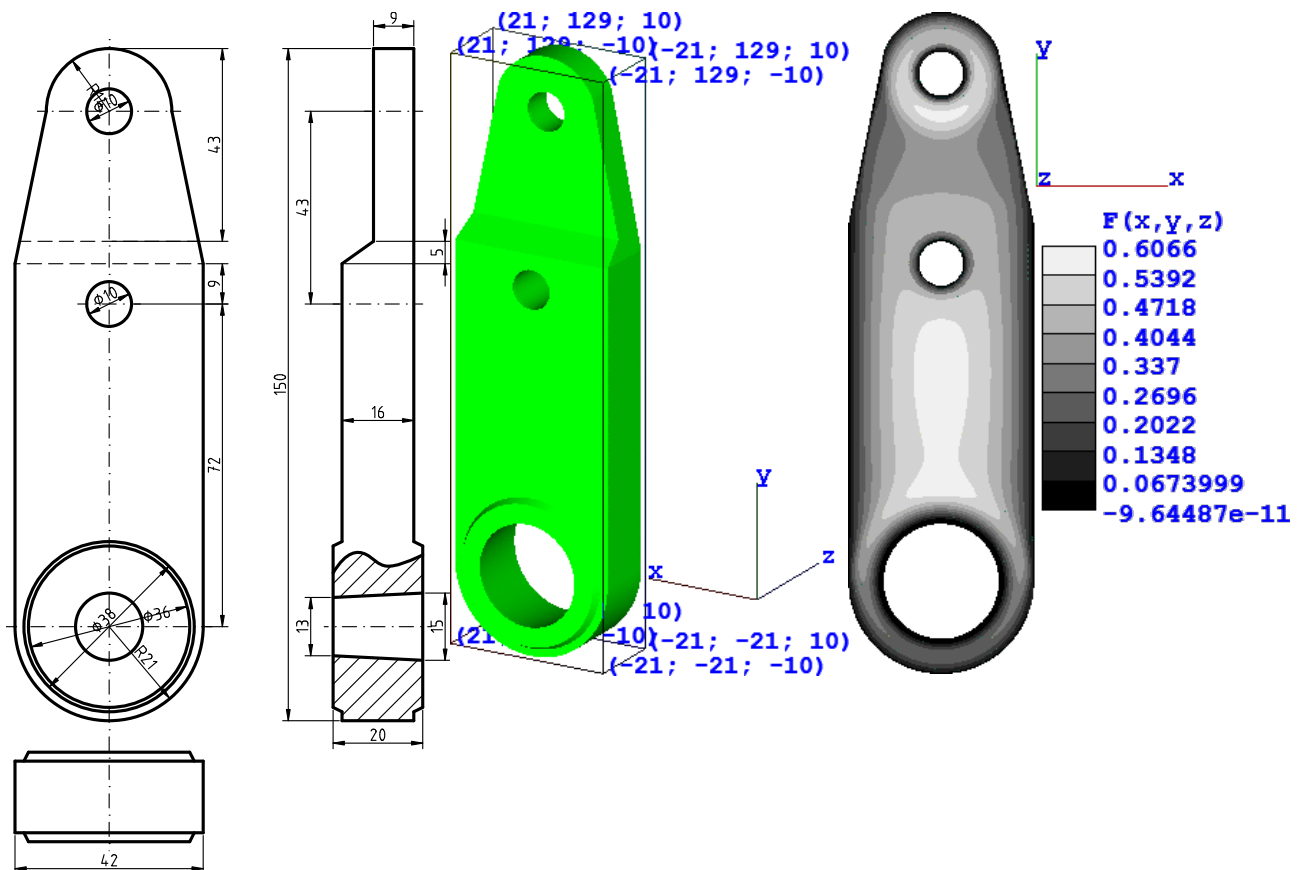


Рисунок 1.38 – Подовжувач важеля передньої підвіски: загальний вигляд і розподіл значень функції в перерізі площиною $z=0$

Підсумкова модель подовжувача важеля передньої підвіски буде отримана в результаті додавання відповідних обмежень щодо глибини, а також конічних виступів й отвору:

$$w(x, y, z) = \left[\left[p(x, y) \wedge \text{stripe}(z, 16) \wedge \neg(\text{line}(z, y, [-8; 81], [-1; 86]) \wedge -z - 1) \right] \vee \right. \\ \left. \vee \left[\text{circle}\left(x, y, 15 + \frac{z}{2}\right) \wedge 10 - z \wedge z - 7 \right] \vee \left[\text{circle}\left(x, y, 15 - \frac{z}{2}\right) \wedge z + 10 \wedge -z - 7 \right] \right] \wedge \quad (1.32) \\ \wedge \neg \text{circle}\left(x, y, 14 + \frac{z}{10}\right).$$

4. Пробка.

Пробка (рис. 1.39) являє собою циліндр змінного радіуса, довжина твірної якого дорівнює 35 мм. Уздовж твірної пробки розташований циліндричний виріз з перетином шестикутної форми.

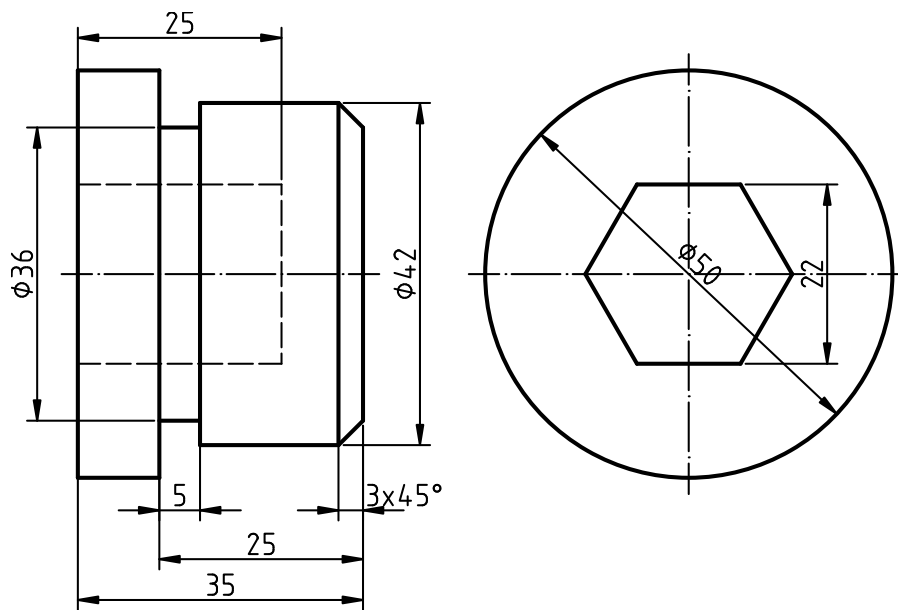


Рисунок 1.39 – Креслення пробки

Модель пробки (рис. 1.40) можна побудувати, «вирізаючи» отвори необхідної форми з циліндра, довжина якого дорівнює 35 мм, а радіус – 50 мм:

$$w(x, y, z) = \text{cylinder}(x, y, z, 25, 35) \wedge \text{groove}(x, y, z + 5, 18, 5) \wedge \\ \wedge \text{groove}(x; y; z - 18, 5; 21; 20) \wedge \text{circle}(x; y; 35, 5 - z) \wedge \quad (1.33) \\ \wedge \neg \left[\text{regular}\left(x, y, \frac{22}{\sqrt{3}}, 6\right) \wedge (7, 5 - z) \right],$$

де cylinder – функція (1.25);

circle – функція (1.11);

regular – функція (1.13);

groove – функція, яка подає кільцеву канавку радіуса r і ширини w :

$$\text{groove}(x, y, z, r, w) = \text{circle}(x, y, r) \vee \neg \text{stripe}(z, w). \quad (1.34)$$

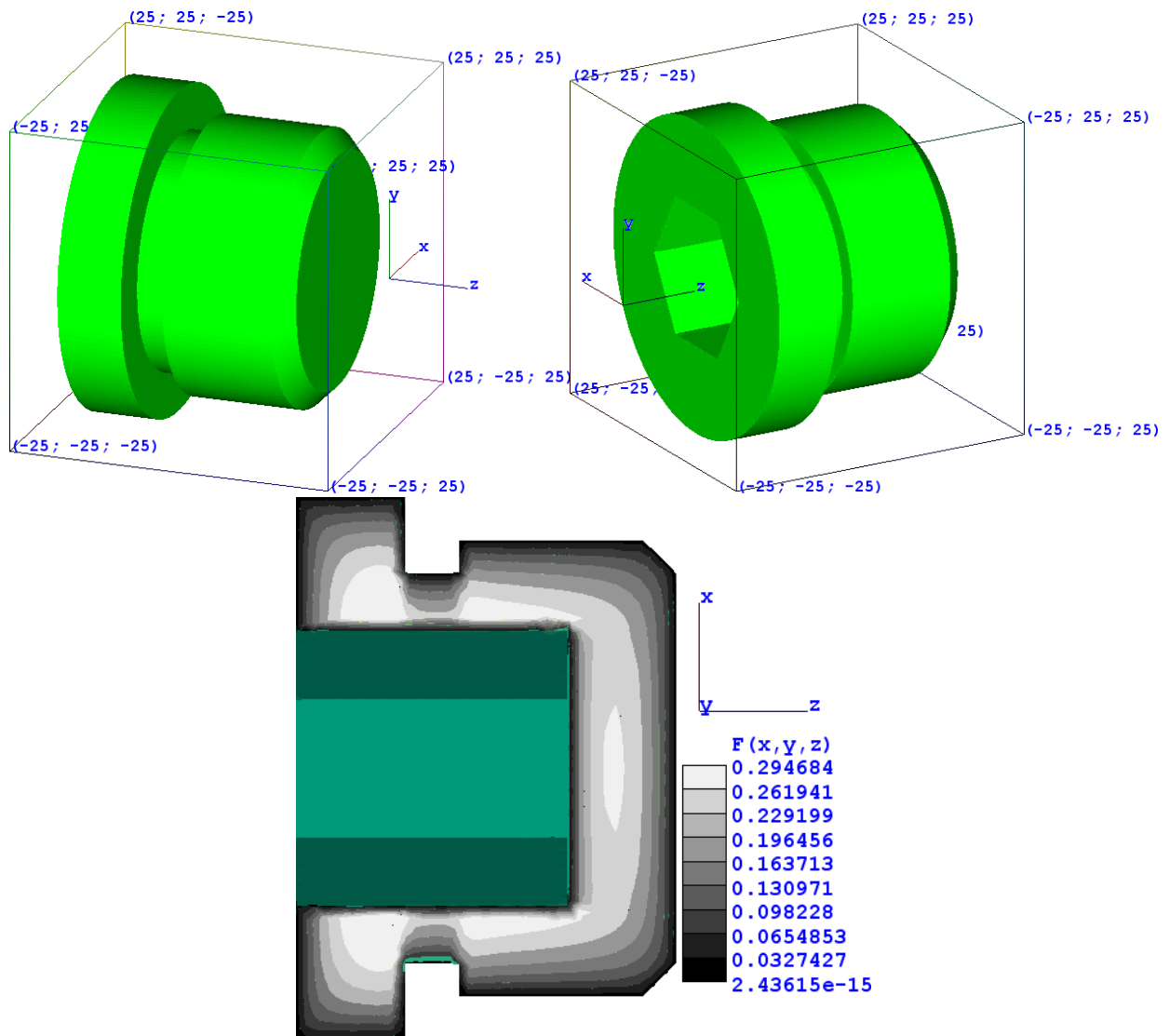


Рисунок 1.40 – Модель пробки: загальний вигляд і розподіл значень функції в перерізі площиною $y=0$

5. Поршень.

Поршень, аналогічно розглянутій вище моделі пробки, є циліндричним тілом (рис. 1.41). У ньому є три кільцеві канавки, які можна подати за допомогою функції *groove*, що визначена формулою (1.34). Поршень володіє двома осьовими вирізами: 1) діаметра 69,3 мм ($75,9-6,6=69,3$) і глибиною 53 мм; 2) діаметра 66,9 мм і глибиною 74 мм ($79,7-5,7=74$). Стінки поршня потовщуються в площині кріплення шатуна. В результаті отримано таку модель (рис. 1.41):

$$\begin{aligned} \text{piston}(x, y, z) = & \text{cylinder}(x, y, z, R, L) \wedge \neg \text{rectangle}_r\left(x, z + \frac{L}{2}, R - a, 2h, r\right) \wedge \\ & \wedge \neg \left[\left\{ \text{hzcylinder}\left(x, y, z + \frac{L}{2} - d_1, R_1\right) \vee \text{hzcylinder}\left(x, y, z + \frac{L}{2} - d_2, R_2\right) \right\} \wedge \text{stripe}(y, w) \right] \wedge \\ & \wedge \neg \text{circle}\left(x, z + \frac{L}{2} - d_1 + R_c, R_c\right) \wedge \text{groove}(x, y, z + \delta_1, R_1, rw_1) \wedge \\ & \wedge \text{groove}(x, y, z + \delta_2, R_1, rw_2) \wedge \text{groove}(x, y, z + \delta_3, R_1, rw_3), \end{aligned}$$

де $R = \frac{75,9}{2}$ мм – радіус поршня;

$L = 79,7$ мм – довжина поршня;

$a = 16$ мм – відстань до прямокутного вирізу на «спідниці» поршня;

$h = 8$ мм – глибина прямокутного вирізу на «спідниці» поршня;

$r = 4$ мм – радіус округлення кутів прямокутника на «спідниці» поршня;

$R_1 = \frac{69,3}{2}$ мм – радіус першого вісєвого вирізу;

$d_1 = 53$ мм – глибина першого вісєвого вирізу;

$R_2 = \frac{66,9}{2}$ мм – радіус другого вісєвого вирізу;

$d_2 = 74$ мм – глибина другого вісєвого вирізу;

$w = 50$ мм – ширина потовщення «спідниці» поршня;

$R_c = \frac{22,2}{2}$ мм – радіус поперечного кругового вирізу;

$\delta_1 = \frac{L}{2} - 57,6 - \frac{rw_1}{2}$ – відстань до центра першої кільцевої канавки;

$rw_1 = 4,2$ мм – ширина першої кільцевої канавки;

$\delta_2 = \delta_1 - \frac{rw_1}{2} - 3 - \frac{rw_2}{2}$ – відстань до центра другої кільцевої канавки;

$rw_2 = 2,2$ мм – ширина другої кільцевої канавки;

$\delta_3 = \delta_2 - \frac{rw_2}{2} - 3 - \frac{rw_3}{2}$ – відстань до центра третьої кільцевої канавки;

$rw_3 = 1,8$ мм – ширина третьої кільцевої канавки;

cylinder – функція (1.25);

rectangle_r – функція (1.16);

hzcylinder(x, y, z, r) = circle(x, y, r) $\wedge \neg z$ – функція, яка подає нескінченний циліндр з однією основою ($z = 0$) і віссю $-Oz$;

circle – функція (1.11);

groove – функція (1.34).

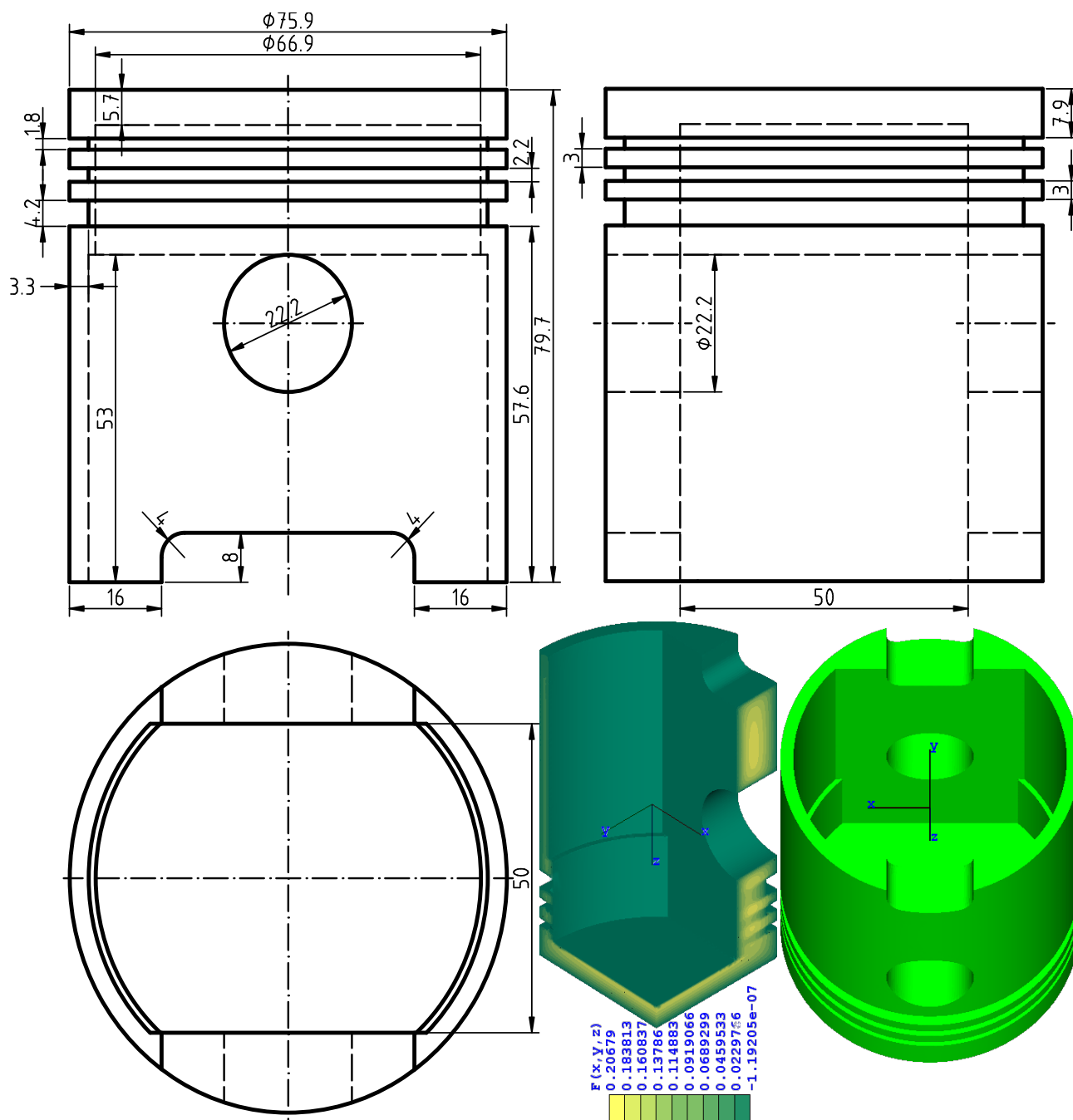


Рисунок 1.41 – Креслення та загальний вигляд поршня

1.7. Гібридний підхід до подання форм об'єктів машинобудування

Функціональне подання з використанням неявних функцій є досить універсальним і дозволяє описувати довільні форми об'єктів. Проте, в авіа-, авто- і суднобудуванні дуже поширене використання параметричних кривих або поверхонь, які обмежують об'єкт (граничне подання). За такої умови, як зазначалося раніше, побудова системи меж для нестандартного об'єкта (наприклад, зі розгалуженою структурою технологічних отворів) також досить

складна. Можливий спосіб спрощення процесу моделювання форм об'єктів – комбінування елементів функціонального та граничного подань (гібридне подання) шляхом перетворення граничних моделей в функціональні. Перехід від граничного подання примітива (системи параметричних кривих або поверхонь) до еквівалентного функціонального (неявного рівняння) за допомогою розв'язання відповідної системи рівнянь можливий лише в окремих відносно простих випадках і, отже, актуальною є розробка методів розв'язання цієї задачі.

В основі граничного подання покладено теорему Жордана-Брауера (узагальнення теореми Жордана для n -мірного простору), яка гарантує, що будь-який $(n-1)$ -вимірний підмноговид в $\mathbb{R}^n$, гомеоморфний сфері, розбиває простір на дві зв'язні компоненти та є їх спільною межею. Тобто, у двовимірному випадку плоска проста (яка не має самоперетину) замкнута крива розбиває площину на дві зв'язні компоненти, одну з яких можна вважати внутрішньою, а іншу – зовнішньою.

У тривимірному просторі (аналогічно до двовимірному випадку) проста замкнена поверхня розбиває простір на дві зв'язні компоненти: внутрішню та зовнішню. Отже, можна вважати, що проста обмежена замкнена множина $L \subset \mathbb{R}^n$ подає певну форму об'єкта Ω (L є межею Ω). Одночасно, для будь-якої замкнутої множини $L \in \mathbb{R}^n$ можна побудувати неперервне неявне рівняння [63], найпростіший варіант якого – нормальне рівняння такого вигляду:

$$F(X) \equiv \inf_{Y \in L} \|X - Y\| = 0, \quad X \in \mathbb{R}^n.$$

Функція

$$\text{dist}(X, L) = \inf_{Y \in L} \|X - Y\| \quad (1.35)$$

є відстанню від точки $X \in \mathbb{R}^n$ до найближчої до неї точки множини L . Ця функція у довільно обраній точці X приймає значення, що дорівнює радіусу найбільшої гіперсфери з центром в X , повністю укладеної (вписаної) в L , якщо X – внутрішня точка L , і дорівнює радіусу найменшої гіперсфери з центром в X , що дотична до L , якщо X – зовнішня точка L .

Припустимо, що дана функція

$$\text{sign}(X, L) = \begin{cases} +1, & X \in \Omega, \\ -1, & X \notin \Omega, \end{cases} \quad (1.36)$$

яка дорівнює $+1$ у внутрішніх точках $\mathbf{L}$ і дорівнює -1 у зовнішніх точках $\mathbf{L}$. Відповідно, функція

$$f(X, \mathbf{L}) = \text{sign}(X, \mathbf{L}) \text{dist}(X, \mathbf{L}) \quad (1.37)$$

буде додатною у внутрішніх точках $\mathbf{L}$, дорівнюватиме нулю, якщо $X \in \mathbf{L}$, і від'ємна у зовнішніх точках $\mathbf{L}$. Така функція буде неявно визначати форму об'єкта Ω . Отже, завдання полягає в побудові функцій (1.35) і (1.36) за умови, що $\mathbf{L}$ – подана параметрично замкнена межа Ω .

1.7.1 Двовимірні форми

Нехай проста замкнена межа деякої двовимірної області подана параметрично у такому вигляді:

$$\text{curve}(t) = \mathbf{p} + x(t)\mathbf{i} + y(t)\mathbf{j}, \quad t_0 \leq t \leq t_1, \quad (1.38)$$

де t – скалярний аргумент;

$x(t)$ і $y(t)$ – неперервні функції параметра t ;

$\mathbf{p} = (x_p; y_p)$ – радіус-вектор довільної точки двовимірного простору (точка прив'язки для характерної точки кривої);

$\{\mathbf{i}, \mathbf{j}\}$ – ортонормований базис декартової системи координат двовимірного простору.

Найбільш поширеними на практиці прикладами примітивів-кривих вигляду (1.38) є:

$$\text{segment}(t) = (1-t)\mathbf{p}_0 + t\mathbf{p}_1, \quad 0 \leq t \leq 1, \quad (1.39)$$

$$\text{circle}(t) = \mathbf{c} + r \cos(t)\mathbf{i} + r \sin(t)\mathbf{j}, \quad 0 \leq t \leq 2\pi, \quad 0 < r, \quad (1.40)$$

$$\text{ellipse}(t) = \mathbf{c} + a \cos(t)\mathbf{i} + b \sin(t)\mathbf{j}, \quad 0 \leq t \leq 2\pi, \quad 0 < a, 0 < b, \quad (1.41)$$

$$\text{cbezier}(t) = \sum_{i=0}^n B_i^n(t) \mathbf{p}_i = \sum_{i=0}^n \frac{n!}{i!(n-i)!} t^i (1-t)^{n-i} \mathbf{p}_i, \quad 0 \leq t \leq 1, \quad (1.42)$$

де функція (1.39) визначає відрізок між точками з радіус-векторами $\mathbf{p}_0$ і $\mathbf{p}_1$;

функція (1.40) – коло радіуса r з центром в точці з радіус-вектором $\mathbf{c}$;

функція (1.41) – еліпс з центром у точці з радіус-вектором $\mathbf{c}$ і півсями a і b ;

функція (1.42) – крива Безьє, побудована на послідовності радіус-векторів $\mathbf{p}_0, \dots, \mathbf{p}_n$, які визначають $n+1$ контрольну точку.

Крива $\text{curve}(t)$, також може бути сплайном або складовою. Найпростішим прикладом останньої є ламана, побудована на послідовності радіус-векторів її вершин $\mathbf{p}_0, \dots, \mathbf{p}_n$:

$$\text{segments}(t) = (1-w)\mathbf{p}_i + w\mathbf{p}_{i+1}, \quad w = t - i, \quad i = \lfloor t \rfloor, \quad 0 \leq t \leq n, \quad (1.43)$$

де $\lfloor t \rfloor$ – операція знаходження найбільшого цілого, меншого, ніж t .

Якщо у формулах (1.42) і (1.43) $\mathbf{p}_0 = \mathbf{p}_n$, то відповідні криві будуть замкненими.

Знаходження відстані від довільної точки простору до найближчої точки кривої – задача досить екстенсивна, тому замінімо криву $\text{curve}(t)$ її дискретної моделлю для наближеного обчислення функції (1.35).

Нехай дискретна модель кривої – це плоский граф, поданий списком ребер у такому вигляді:

$$\mathbf{E}_2 = \{e_i = (\mathbf{p}_{i0}; \mathbf{p}_{i1})\}, \quad i = \overline{1, n},$$

де $\mathbf{p}_{i0}$ і $\mathbf{p}_{i1}$ – відповідно, координати (радіус-вектори) початку та кінця ребра e_i ; n – кількість ребер.

У такому випадку функція (1.35) отримає вигляд

$$\text{dist}(x, y, \text{curve}(t)) = \text{dist}(\mathbf{p}, \text{curve}(t)) \approx \text{dist}_2(\mathbf{p}, \mathbf{E}_2) = \min_{e_i = (\mathbf{p}_{i0}; \mathbf{p}_{i1}) \in \mathbf{E}_2} d_2(\mathbf{p}, \mathbf{p}_{i0}, \mathbf{p}_{i1}), \quad (1.44)$$

де $\mathbf{p} = (x; y) \in \mathbb{R}^2$ – довільна точка площини;

$d_2(\mathbf{p}, \mathbf{p}_{i0}, \mathbf{p}_{i1})$ – функція для визначення відстані від точки $\mathbf{p}$ до відрізка, заданого точками $\mathbf{p}_{i0}$ і $\mathbf{p}_{i1}$ (вершини ребра $e_i = (\mathbf{p}_{i0}; \mathbf{p}_{i1})$).

Функцію $d_2(\mathbf{p}, \mathbf{p}_{i0}, \mathbf{p}_{i1})$ можна задати такою формулою:

$$d_2(\mathbf{p}, \mathbf{p}_{i0}, \mathbf{p}_{i1}) = \begin{cases} \|\mathbf{p} - \mathbf{p}_{i0}\|, & \text{якщо } (\mathbf{p} - \mathbf{p}_{i0}) \cdot (\mathbf{p}_{i1} - \mathbf{p}_{i0}) \leq 0, \\ \|\mathbf{p} - \mathbf{p}_{i1}\|, & \text{якщо } (\mathbf{p}_{i1} - \mathbf{p}_{i0}) \cdot (\mathbf{p}_{i1} - \mathbf{p}_{i0}) \leq (\mathbf{p} - \mathbf{p}_{i0}) \cdot (\mathbf{p}_{i1} - \mathbf{p}_{i0}), \\ \left\| \mathbf{p} - \left(\mathbf{p}_{i0} + \frac{(\mathbf{p} - \mathbf{p}_{i0}) \cdot (\mathbf{p}_{i1} - \mathbf{p}_{i0})}{(\mathbf{p}_{i1} - \mathbf{p}_{i0}) \cdot (\mathbf{p}_{i1} - \mathbf{p}_{i0})} (\mathbf{p}_{i1} - \mathbf{p}_{i0}) \right) \right\|, & \text{інакше.} \end{cases} \quad (1.45)$$

Варто відзначити, що функцію (1.45) також можна використовувати для знаходження відстані від точки до відрізка в тривимірному випадку [281]. Вочевидь зі збільшенням числа ребер в графі $\mathbf{E}_2$ точність формули (1.44) буде збільшуватися. Однак кривина кривої $\text{curve}(t)$ може бути змінною, отже, для

досягнення найкращої апроксимації необхідно забезпечити згущення ребер у області найбільшої кривини. У такому випадку можна скористатися наступним алгоритмом.

algorithm nonuniform-curve-mesh

input:

$\text{curve}(t)$, $t_0 \leq t \leq t_1$ – формула кривої, що описує границю області

$n \geq 2$ – кількість вузлів в початковій дискретній моделі контуру

δ – максимальне відхилення середини ребра від границі

output:

E_2 – дискретна модель кривої

begin

$E_t \leftarrow \{e_i = (t_{i0}; t_{i1})\}, t_{i0} = t_0 + ih, t_{i1} = t_0 + (i+1)h, h = \frac{t_1 - t_0}{n-1}, i = \overline{0, n-2}$

repeat

$f \leftarrow \text{false}$

for each $e_i \in E_t$ **do**

if $\frac{\left\| \frac{\text{curve}(t_{i0}) + \text{curve}(t_{i1})}{2} - \text{curve}\left(\frac{t_{i0} + t_{i1}}{2}\right) \right\|}{\|\text{curve}(t_{i1}) - \text{curve}(t_{i0})\|} > \delta$ **then**

$E_t \leftarrow E_t \setminus e_i$

$E_t \leftarrow E_t \cup \left(t_{i0}; \frac{t_{i0} + t_{i1}}{2}\right)$

$E_t \leftarrow E_t \cup \left(\frac{t_{i0} + t_{i1}}{2}; t_{i1}\right)$

$f \leftarrow \text{true}$

end if

end for

until $f = \text{true}$

$E_2 \leftarrow \{e_i = (\text{curve}(t_{i0}); \text{curve}(t_{i1}))\}, i = \overline{1, |E_t|}$

return E_2

У результаті буде побудована адаптивна дискретна модель межі області, яку можна використовувати для апроксимації за формулою (1.44). Наприклад, область, яка обмежена замкнутою кривою Безьє, що побудована на послідовності з 10 контрольних точок: $\mathbf{p}_0 = (0; 0)$, $\mathbf{p}_1 = (0; 3)$, $\mathbf{p}_2 = (1; 3)$, $\mathbf{p}_3 = (1; 1)$, $\mathbf{p}_4 = (4; 1)$, $\mathbf{p}_5 = (4; -1)$, $\mathbf{p}_6 = (1; -1)$, $\mathbf{p}_7 = (1; -3)$, $\mathbf{p}_8 = (0; -3)$ і $\mathbf{p}_9 = (0; 0)$, подається дискретною моделлю, зображеною на рис. 1.42 (вважається, що $t_0 = 0$, $t_1 = 1$, $n = 41$ і $\delta = 0,02$).

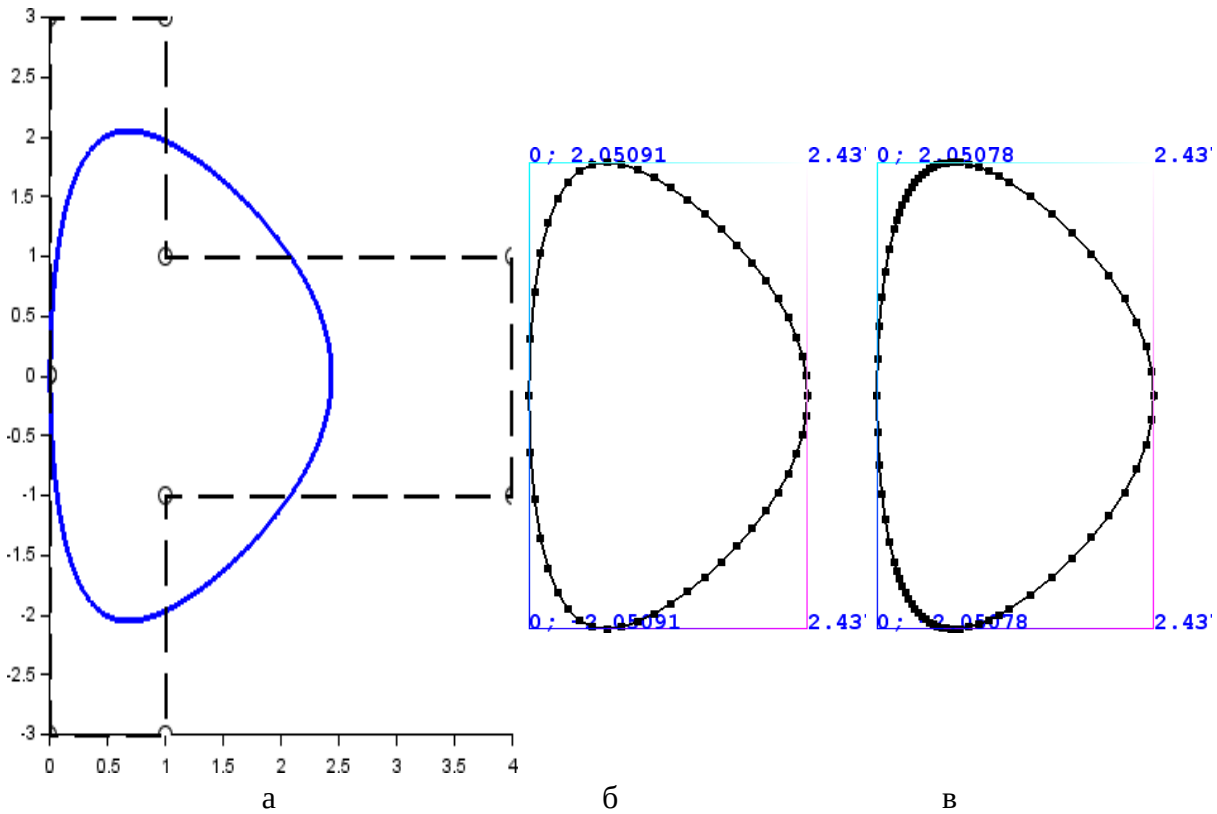


Рисунок 1.42 – Замкнена крива Безьє: а – вихідна крива та її контрольні точки; б – початкова дискретна модель контура; в – адаптивна дискретна модель контура

Граф E_2 , власне, є замкнутим багатокутником, який апроксимує криву. Відповідно до теореми Жордана для побудови функції (1.36) можна скористатися тестом парності, можливо, перша реалізація якого розглянута в роботі [282]. Він полягає в тому, що з точки P випускають промінь, наприклад, у додатному напрямку осі Ox . Далі для визначення належності точки багатокутника досить підрахувати число s – кількість перетинів променя та ребер графа. Єдиний вироджений випадок буде, якщо промінь перетинає ребра у вершині. У такому випадку на практиці використовують такий прийом: ребро вважають пересіченим, якщо одна його вершина строго нижче променя, а друга вище або лежить на промені. У результаті функція (1.36) отримає такий вигляд:

$$\text{sign}(x, y, \text{curve}(t)) = \text{sign}(\mathbf{p}, \text{curve}(t)) \approx \text{sign}_2(\mathbf{p}, E_2) = \begin{cases} +1, & s \text{ парне,} \\ -1, & s \text{ непарне.} \end{cases} \quad (1.46)$$

Отже, абсолютне значення функції (1.37) можна обчислити з використанням формул (1.44) і (1.45) з деякою точністю. Точність визначають параметри метода: кількістю точок у початковій дискретній моделі контуру n і параметр апроксимації δ . Водночас знак цієї функції можна обчислити за допомогою тесту парності (1.46).

Наприклад, для замкнутої кривої Безьє, зображеної на рис. 1.42, розподіл значень апроксимації функції (1.37), якщо $n=41$ і $\delta=0,02$, матиме вигляд, наведений на рис. 1.43.

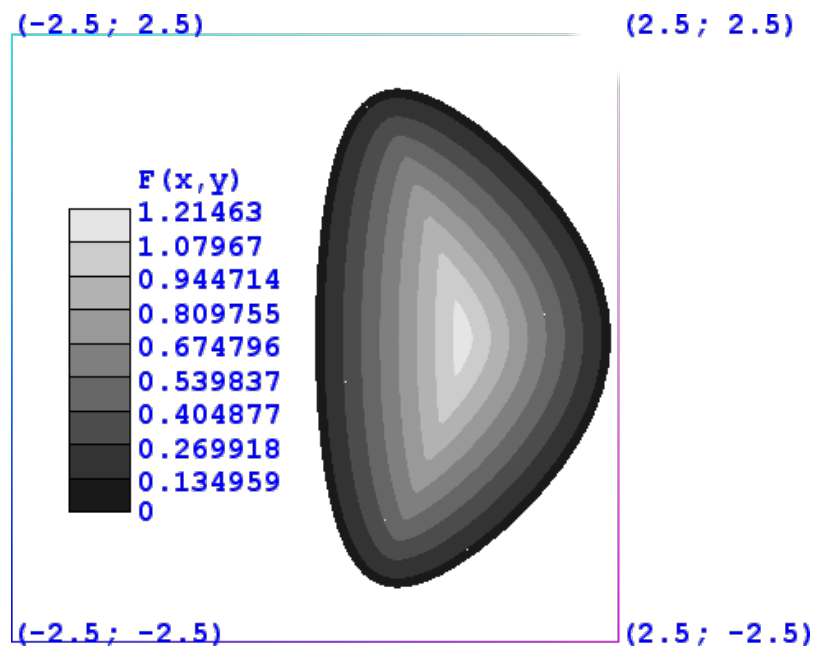


Рисунок 1.43 – Розподіл значень функції (1.37) в області, яка обмежена замкненою кривою Безьє

Якщо вихідною кривою буде, наприклад, багатокутник, визначений замкненою ламаною (1.43) з послідовністю вершин $\mathbf{p}_0=(0; 0)$, $\mathbf{p}_1=(3; 0)$, $\mathbf{p}_2=(3; 2)$, $\mathbf{p}_3=(2; 2)$, $\mathbf{p}_4=(2; 1)$, $\mathbf{p}_5=(1; 1)$, $\mathbf{p}_6=(1; 4)$, $\mathbf{p}_7=(2; 4)$, $\mathbf{p}_8=(2; 3)$, $\mathbf{p}_9=(3; 3)$, $\mathbf{p}_{10}=(3; 5)$, $\mathbf{p}_{11}=(0; 5)$, $\mathbf{p}_{12}=(0; 0)$, $t_0=0$, $t_1=12$ і $n=13$ (значення параметра δ у даному випадку не впливає на результат), то розподіл значень функції (1.37) матиме вигляд, наведений на рис. 1.44.

У роботах [283–285] розглянуті особливості моделювання профілів крил з використанням кривих Безьє. Отримані у цих роботах результати дозволяють досліджувати аеродинамічні властивості, але у задачах дослідження міцності необхідно враховувати також технологічні отвори. Наприклад, симетричний профіль NASA 0011 [283] (рис. 1.45) можна подати складеною кривою

$$\text{наса}_{0011}(t) = \begin{cases} \sum_{i=0}^n \frac{n!}{i!(n-i)!} t^i (1-t)^{n-i} \mathbf{t}_i, & 0 \leq t \leq 1, \\ \sum_{i=0}^n \frac{n!}{i!(n-i)!} (t-1)^i (2-t)^{n-i} \mathbf{b}_i, & 1 < t \leq 2, \end{cases}$$

де $n=5$ – степені кривих;

$t_0=(0; 0)$, $t_1=(1,14034; 2,44284)$, $t_2=(11,1025; 3,65375)$, $t_3=(30,8809; 1,13309)$, $t_4=(39,9276; 0,09712)$, $t_5=(40; 0)$ – контрольні точки кривої Безьє п'ятого степеня, які відповідають верхньому вигину крила;

$b_0=(0; 0)$, $b_1=(1,14034; -2,44284)$, $b_2=(11,1025; -3,65375)$, $b_3=(30,8809; -1,13309)$, $b_4=(39,9276; -0,09712)$, $b_5=(40; 0)$ – контрольні точки кривої Безьє п'ятого степеня, які відповідають нижньому вигину крила.

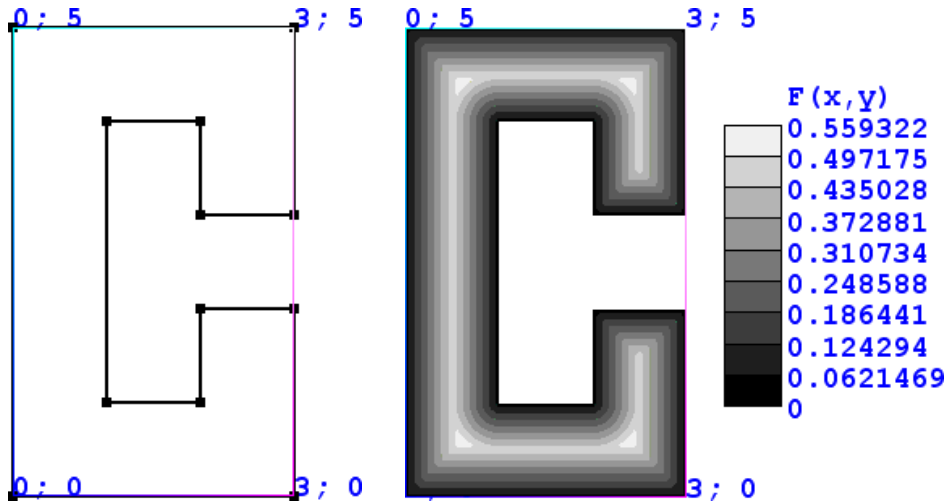


Рисунок 1.44 – Розподіл значень функції (1.37) для багатокутника

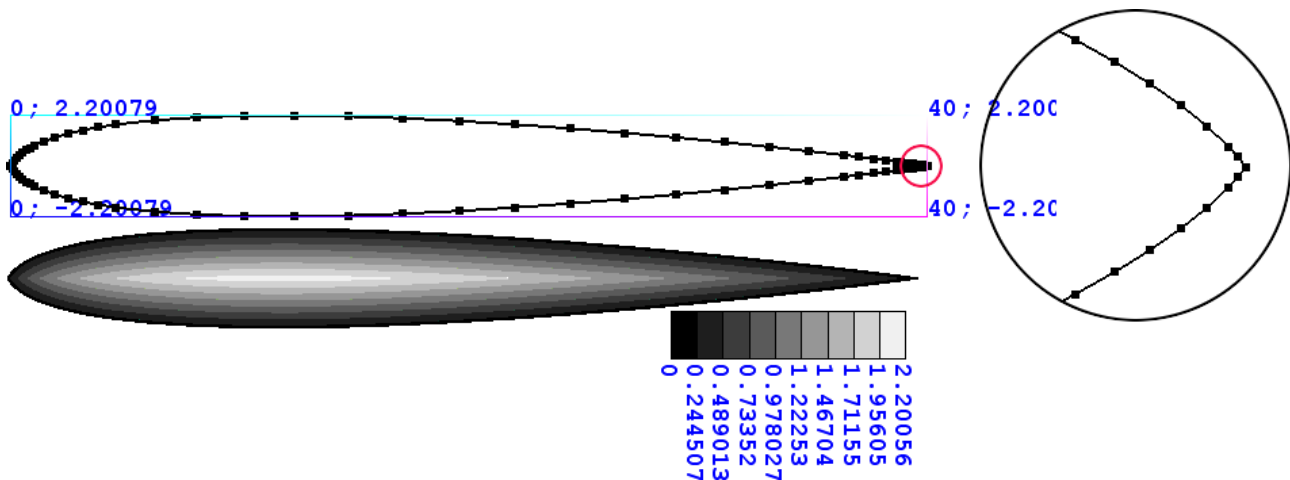


Рисунок 1.45 – Профіль NASA 0011: дискретна модель контуру та розподіл значень функції (1.37)

Варто зазначити, що функції вигляду (1.37), побудовані для контурів двовимірних областей, можна використовувати як операнди (примітиви) R-операцій разом з розглянутими вище неявними функціями. Наприклад, профіль крила з трьома технологічними отворами (рис. 1.46) можна подати такою функцією:

$$f_{naca}(x, y) = f(x, y, naca_{0011}(t)) \wedge \neg \text{rectangle}(x - x_1, y, w_1, h_1) \wedge \neg \text{circle}(x - x_2, y, r_2) \wedge \neg \text{rectangle}(x - x_3, y, w_3, h_3),$$

де circle – функція (1.11);

rectangle – функція (1.14);

$x_1, x_2, x_3, w_1, h_1, r_2, w_3$ – параметри положення та розмірів отворів.

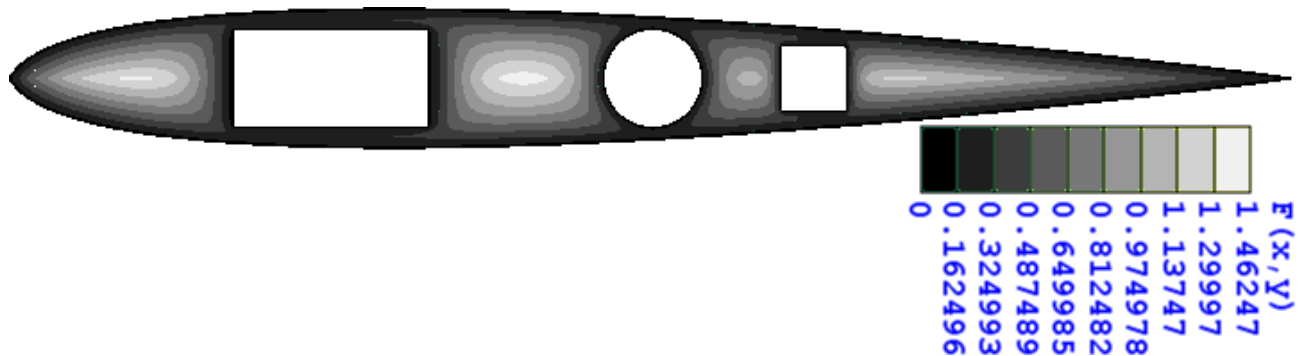


Рисунок 1.46 – Профіль NASA 0011 з технологічними отворами, якщо $x_1=10, x_2=20, x_3=25, w_1=6, h_1=3, r_2=1,5, w_3=2$

1.7.2 Тривимірні форми

Нехай проста замкнена межа деякої тривимірної області задана m частинами (клаптями, патчами, англійською – patches), які є параметричними поверхнями:

$$\begin{aligned} \text{surface}_l(u, v) &= \mathbf{q} + x(u, v)\mathbf{i} + y(u, v)\mathbf{j} + z(u, v)\mathbf{k}, \\ u_{l0} \leq u \leq u_{l1}, \quad v_{l0} \leq v \leq v_{l1} \quad l &= \overline{1, m}, \end{aligned} \quad (1.47)$$

де u і v – скалярні аргументи;

$x(u, v), y(u, v), z(u, v)$ – неперервні функції параметрів u і v ;

$\mathbf{q}=(x_q; y_q; z_q)$ – радіус-вектор довільної точки тривимірного простору (точки прив'язки характерної точки поверхні);

$\{\mathbf{i}, \mathbf{j}, \mathbf{k}\}$ – ортонормований базис декартової системи координат тривимірного простору.

Як у двовимірному випадку, частини $\text{surface}_j(u, v)$, також, можуть бути сплайнами або складеними поверхнями Безьє.

Для знаходження відстані від довільної точки простору до найближчої точки поверхні побудуємо дискретну модель межі області, використовуючи трикутні елементи. До такої дискретної моделі висувається тільки вимога узгодженості вузлів (і ребер в порядку їх обходу) елементів.

Нехай дискретна модель поверхні – це множина, що складається з трикутних елементів, нормаль до яких (в порядку обходу вузлів) спрямована назовні об'єкта:

$$\mathbf{E}_3 = \{tr_i = (\mathbf{q}_{i0}; \mathbf{q}_{i1}; \mathbf{q}_{i2})\}, \quad i = \overline{1, n},$$

де $\mathbf{q}_{i0}$, $\mathbf{q}_{i1}$ і $\mathbf{q}_{i2}$ – координати вузлів трикутника tr_i , впорядковані проти годинникової стрілки.

У такому випадку при наближеному обчисленні формули (1.35) границю $\mathbf{L}$ можна замінити дискретною моделлю $\mathbf{E}_3$:

$$\text{dist}(x, y, z, \mathbf{L}) = \text{dist}(\mathbf{q}, \mathbf{L}) \approx \text{dist}_3(\mathbf{q}, \mathbf{E}_3) = \min_{t_i = (\mathbf{q}_{i0}; \mathbf{q}_{i1}; \mathbf{q}_{i2}) \in \mathbf{E}_3} d_3(\mathbf{q}, \mathbf{q}_{i0}, \mathbf{q}_{i1}, \mathbf{q}_{i2}), \quad (1.48)$$

де $\mathbf{q} = (x, y, z) \in \mathbf{R}^3$ – радіус-вектор довільної точки простору;

$d_3(\mathbf{q}, \mathbf{q}_{i0}, \mathbf{q}_{i1}, \mathbf{q}_{i2})$ – функція для визначення відстані від точки $\mathbf{q}$ до трикутника, заданого точками $\mathbf{q}_{i0}$, $\mathbf{q}_{i1}$ і $\mathbf{q}_{i2}$.

Функцію $d_3(\mathbf{q}, \mathbf{q}_{i0}, \mathbf{q}_{i1}, \mathbf{q}_{i2})$, яка використана у формулі (1.48), можна обчислити за допомогою такої формули:

$$d_3(\mathbf{q}, \mathbf{q}_{i0}, \mathbf{q}_{i1}, \mathbf{q}_{i2}) = \begin{cases} \|l\|, & v_0 \geq 0 \wedge v_1 \geq 0 \wedge v_2 \geq 0, \\ \|\mathbf{q} - \mathbf{q}_{i0}\|, & v_0 < 0 \wedge v_1 \geq 0 \wedge v_2 < 0, \\ \|\mathbf{q} - \mathbf{q}_{i1}\|, & v_0 < 0 \wedge v_1 < 0 \wedge v_2 \geq 0, \\ \|\mathbf{q} - \mathbf{q}_{i2}\|, & v_0 \geq 0 \wedge v_1 < 0 \wedge v_2 < 0, \\ d_2(\mathbf{q}, \mathbf{q}_{i0}, \mathbf{q}_{i1}), & v_0 < 0 \wedge v_1 \geq 0 \wedge v_2 \geq 0, \\ d_2(\mathbf{q}, \mathbf{q}_{i1}, \mathbf{q}_{i2}), & v_0 \geq 0 \wedge v_1 < 0 \wedge v_2 \geq 0, \\ d_2(\mathbf{q}, \mathbf{q}_{i2}, \mathbf{q}_{i0}), & v_0 \geq 0 \wedge v_1 \geq 0 \wedge v_2 < 0, \end{cases} \quad (1.49)$$

де $v_0 = \mathbf{n} \cdot (\mathbf{e}_{01} \times \mathbf{c}_0)$, $v_1 = \mathbf{n} \cdot (\mathbf{e}_{12} \times \mathbf{c}_1)$ і $v_2 = \mathbf{n} \cdot (\mathbf{e}_{20} \times \mathbf{c}_2)$ – значення, які використовуються для визначення взаємного розташування точки і вершин трикутника;

$\mathbf{e}_{01} = \mathbf{q}_{i1} - \mathbf{q}_{i0}$, $\mathbf{e}_{12} = \mathbf{q}_{i2} - \mathbf{q}_{i1}$ і $\mathbf{e}_{20} = \mathbf{q}_{i0} - \mathbf{q}_{i2}$ – вектори, утворені сторонами трикутника;

$\mathbf{n} = \mathbf{e}_{01} \times (-\mathbf{e}_{20})$ – зовнішня нормаль до площини трикутника;

$l = \frac{(\mathbf{q} - \mathbf{q}_{i0}) \cdot \mathbf{n}}{\|\mathbf{n}\|}$ – відстань від $\mathbf{q}$ до площини трикутника;

$\mathbf{b} = \mathbf{q} - \frac{l}{\|\mathbf{n}\|} \mathbf{n}$ – радіус-вектор проекції $\mathbf{q}$ на площину трикутника;

$\mathbf{c}_0 = \mathbf{b} - \mathbf{q}_{i0}$, $\mathbf{c}_1 = \mathbf{b} - \mathbf{q}_{i1}$ і $\mathbf{c}_2 = \mathbf{b} - \mathbf{q}_{i2}$ – вектори, що з'єднують вершини трикутника з вершиною радіус-вектора $\mathbf{b}$.

З точки зору раціональності використання комп'ютерних ресурсів, необхідних для обчислення формули (1.48), як і у двовимірному випадку, граф E_3 повинен бути адаптивним до кривини границі. Побудову графа для множини параметричних клаптів можна виконати за допомогою такого алгоритму:

algorithm nonuniform-surface-mesh

input:

M – кількість частин поверхні

$\text{surface}_l(u, v), u_{l0} \leq u \leq u_{l1}, v_{l0} \leq v \leq v_{l1}, l = \overline{1, M}$ – формула поверхні кожного клаптя

$n_l \geq 2$ – кількість вузлів в початковій дискретної моделі контуру уздовж абсциси параметричних координат

$m_l \geq 2$ – кількість вузлів в початковій дискретної моделі контуру уздовж ординати параметричних координат

δ – максимальне відхилення середини ребра від границі

output:

E_3 – дискретна модель границі

begin

$E_3 \leftarrow \emptyset$

for $l \leftarrow 1$ **to** M **do**

for $i \leftarrow 0$ **to** n_l **do**

for $j \leftarrow 0$ **to** m_l **do**

$$p_{ij} \leftarrow \left(u_{l0} + i \frac{u_{l1} - u_{l0}}{n_l - 1}; v_{l0} + j \frac{v_{l1} - v_{l0}}{m_l - 1} \right)$$

end for

end for

$E_{3l} \leftarrow \emptyset$

for $i \leftarrow 0$ **to** $n_l - 1$ **do**

for $j \leftarrow 0$ **to** $m_l - 1$ **do**

if $p_{ij} \neq p_{i+1j}$ **and** $p_{ij} \neq p_{i+1j}$ **and** $p_{i+1j} \neq p_{i+1j+1}$ **then**

$E_{3l} \leftarrow E_{3l} \cup (p_{ij}; p_{i+1j}; p_{i+1j+1})$

end if

if $p_{ij} \neq p_{i+1j+1}$ **and** $p_{ij} \neq p_{ij+1}$ **and** $p_{ij+1} \neq p_{i+1j+1}$ **then**

$E_{3l} \leftarrow E_{3l} \cup (p_{ij}; p_{i+1j+1}; p_{ij+1})$

end if

end for

end for

repeat

$f \leftarrow \text{false}$

for each $t = (p_0; p_1; p_2) \in E_{3l}$ **do**

```

if  $\left\| \frac{\text{surface}_l(\mathbf{p}_0) + \text{surface}_l(\mathbf{p}_1)}{2} - \text{surface}_l\left(\frac{\mathbf{p}_0 + \mathbf{p}_1}{2}\right) \right\| > \delta$  then
     $\mathbf{E}_{3l} \leftarrow \mathbf{E}_{3l} \setminus t$ 
     $\mathbf{E}_{3l} \leftarrow \mathbf{E}_{3l} \cup \left( \mathbf{p}_0; \frac{\mathbf{p}_0 + \mathbf{p}_1}{2}; \mathbf{p}_2 \right)$ 
     $\mathbf{E}_{3l} \leftarrow \mathbf{E}_{3l} \cup \left( \mathbf{p}_2; \frac{\mathbf{p}_0 + \mathbf{p}_1}{2}; \mathbf{p}_1 \right)$ 
     $f \leftarrow \text{true}$ 
else if  $\left\| \frac{\text{surface}_l(\mathbf{p}_1) + \text{surface}_l(\mathbf{p}_2)}{2} - \text{surface}_l\left(\frac{\mathbf{p}_1 + \mathbf{p}_2}{2}\right) \right\| > \delta$  then
     $\mathbf{E}_{3l} \leftarrow \mathbf{E}_{3l} \setminus t$ 
     $\mathbf{E}_{3l} \leftarrow \mathbf{E}_{3l} \cup \left( \mathbf{p}_1; \frac{\mathbf{p}_1 + \mathbf{p}_2}{2}; \mathbf{p}_0 \right)$ 
     $\mathbf{E}_{3l} \leftarrow \mathbf{E}_{3l} \cup \left( \mathbf{p}_0; \frac{\mathbf{p}_1 + \mathbf{p}_2}{2}; \mathbf{p}_2 \right)$ 
     $f \leftarrow \text{true}$ 
else if  $\left\| \frac{\text{surface}_l(\mathbf{p}_2) + \text{surface}_l(\mathbf{p}_0)}{2} - \text{surface}_l\left(\frac{\mathbf{p}_2 + \mathbf{p}_0}{2}\right) \right\| > \delta$  then
     $\mathbf{E}_{3l} \leftarrow \mathbf{E}_{3l} \setminus t$ 
     $\mathbf{E}_{3l} \leftarrow \mathbf{E}_{3l} \cup \left( \mathbf{p}_2; \frac{\mathbf{p}_2 + \mathbf{p}_0}{2}; \mathbf{p}_1 \right)$ 
     $\mathbf{E}_{3l} \leftarrow \mathbf{E}_{3l} \cup \left( \mathbf{p}_1; \frac{\mathbf{p}_2 + \mathbf{p}_0}{2}; \mathbf{p}_0 \right)$ 
     $f \leftarrow \text{true}$ 
end if
end for
until  $f = \text{true}$ 
     $\mathbf{E}_{tl} = \{ tr_i = (\text{surface}(\mathbf{p}_{i0}); \text{surface}(\mathbf{p}_{i1}); \text{surface}(\mathbf{p}_{i2})) \}, i = \overline{1, |\mathbf{E}_{3l}|}$ 
     $\mathbf{E}_3 = \mathbf{E}_3 \cup \mathbf{E}_{tl}$ 
end for
return  $\mathbf{E}_3$ 

```

У результаті виконання алгоритму буде побудована адаптивна до кривини поверхні дискретна модель, яку можна використовувати для апроксимації за

формулою (1.48). Деякі приклади адаптації при різних значеннях параметра δ показані на рис. 1.47.

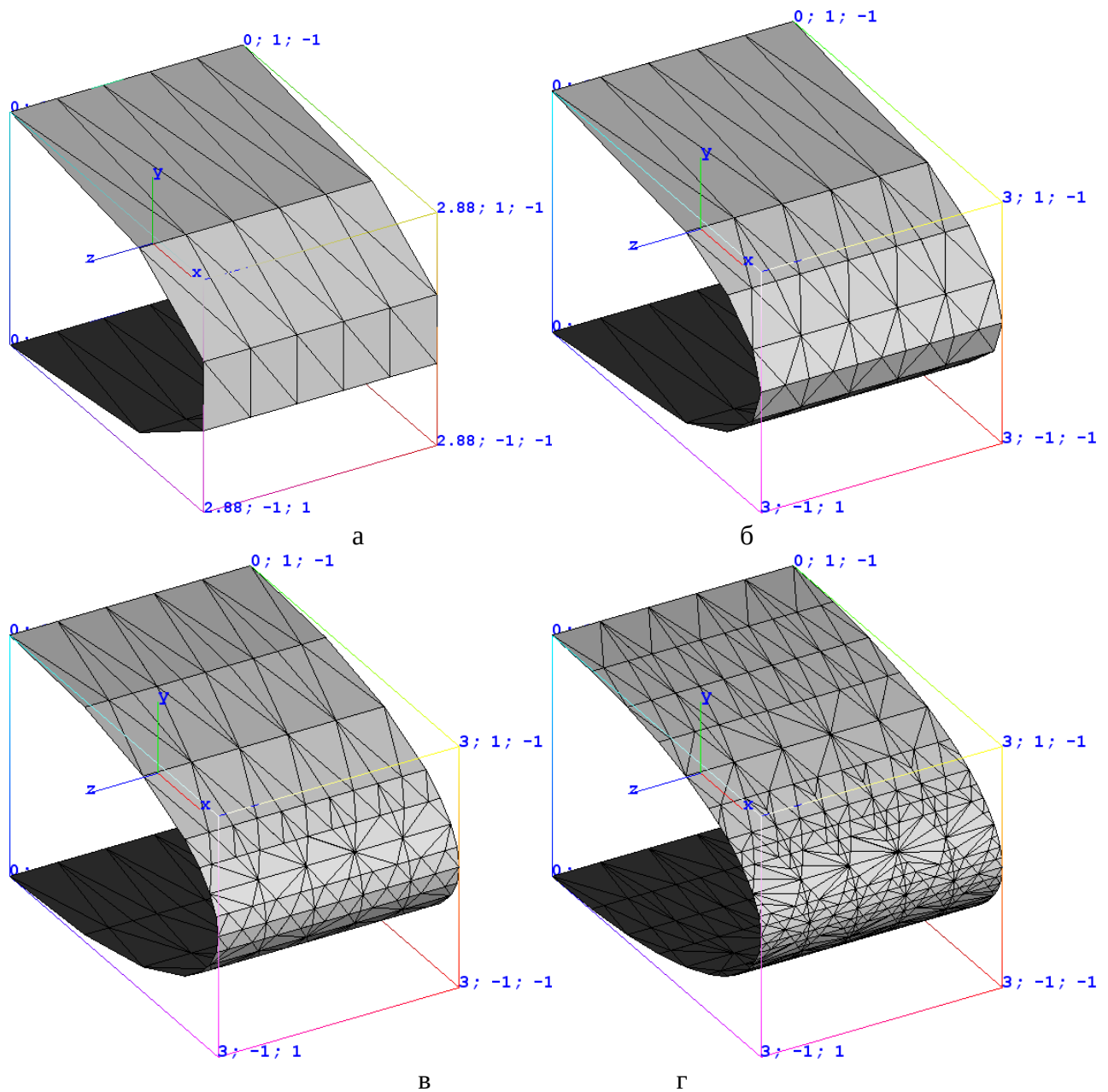


Рисунок 1.47 – Дискретизація поверхні, якщо $n_l=6$ і $m_l=6$: а – без адаптації до кривини; б – адаптація до кривини поверхні, якщо $\delta=0,1$; в – адаптація до кривини поверхні, якщо $\delta=0,05$; г – адаптація до кривини поверхні, якщо $\delta=0,025$

Результатом роботи алгоритму nonuniform-surface-mesh, вхідними даного якого є частини замкненої обмеженої поверхні, буде замкнений багатогранник з трикутними гранями. У цьому багатограннику кількість вузлів і трикутних елементів буде збільшуватися на частинах поверхні з більшою кривиною.

Для визначення знака функції відстані у точці $\mathbf{q}=(x_q; y_q; z_q)$ на першому кроці будуються багатокутники, утворені перетином багатогранника площиною $z=z_q$ (аналогічно двовимірному випадку, для виправлення можливих

вироджених випадків, визначених дотиком площиною до трикутника, вважається, що трикутник перетинається площиною, якщо хоча б один вузол строго глибше та хоча б один ближче або лежить у площині $z=z_q$). Потім використовується процедура підрахунку числа перетинів променя та багатокутника, яка повністю еквівалентна двовимірному випадку. У результаті функція визначення знака набуде вигляду (1.46).

Наприклад, розглянемо поверхню моделі літального апарату (рис. 1.48), визначеного шістьма частинами (поверхнями Безьє) з контрольними точками, заданими матрицями:

$$\begin{aligned} \mathbf{A} &= \begin{bmatrix} (0; -4; 0) & (0; -3,9; 0,01) & (0; -1,1; 0,4) & (0; 0; 3) \\ (4; -2; 0) & (4; -1,9; 0,01) & (4; -0,1; 0,4) & (4; 0; 3) \\ (7; -3; 0) & (7; -2,9; 0,01) & (7; -0,1; 0,4) & (2; 0; 1) \\ (8; -1; 0) & (8; -0,9; 0) & (8; 0; 0) & (8; 0; 0) \end{bmatrix}, \\ \mathbf{B} &= \begin{bmatrix} (8; 1; 0) & (8; 0,9; 0) & (8; 0; 0) & (8; 0; 0) \\ (7; 3; 0) & (7; 2,9; 0,01) & (7; 0,1; 0,4) & (2; 0; 1) \\ (4; 2; 0) & (4; 1,9; 0,01) & (4; 0,1; 0,4) & (4; 0; 3) \\ (0; 4; 0) & (0; 3,9; 0,01) & (0; 1,1; 0,4) & (0; 0; 3) \end{bmatrix}, \\ \mathbf{C} &= \begin{bmatrix} (8; -1; 0) & (8; -0,9; 0) & (8; 0; 0) & (8; 0; 0) \\ (7; -3; 0) & (7; -2,9; -0,01) & (7; -0,1; -0,4) & (7; 0; -0,4) \\ (4; -2; 0) & (4; -1,9; -0,01) & (4; -0,1; -0,4) & (4; 0; -0,4) \\ (0; -4; 0) & (0; -3,9; -0,01) & (0; -1,1; -0,4) & (0; 0; -0,4) \end{bmatrix}, \\ \mathbf{D} &= \begin{bmatrix} (0; 4; 0) & (0; 3,9; -0,01) & (0; 1,1; -0,4) & (0; 0; -0,4) \\ (4; 2; 0) & (4; 1,9; -0,01) & (4; 0,1; -0,4) & (4; 0; -0,4) \\ (7; 3; 0) & (7; 2,9; -0,01) & (7; 0,1; -0,4) & (7; 0; -0,4) \\ (8; 1; 0) & (8; 0,9; 0) & (8; 0; 0) & (8; 0; 0) \end{bmatrix}, \\ \mathbf{E} &= \begin{bmatrix} (0; -4; 0) & (0; -3,9; -0,01) & (0; -1,1; -0,4) & (0; 0; -0,4) \\ (0; -4; 0) & (0; -3,9; 0,01) & (0; -1,1; 0,4) & (0; 0; 3) \end{bmatrix}, \\ \mathbf{F} &= \begin{bmatrix} (0; 4; 0) & (0; 3,9; 0,01) & (0; 1,1; 0,4) & (0; 0; 3) \\ (0; 4; 0) & (0; 3,9; -0,01) & (0; 1,1; -0,4) & (0; 0; -0,4) \end{bmatrix}. \end{aligned}$$

Поверхню кожного клаптя можна описати параметрично такою формулою [43]:

$$\text{sbezier}(u, v) = \sum_{i=0}^n \sum_{j=0}^m B_i^n(u) B_j^m(v) \mathbf{Q}_{ij}, \quad 0 \leq u \leq 1, \quad 0 \leq v \leq 1, \quad (1.50)$$

де $B_i^n(t) = \frac{n!}{i!(n-i)!} t^i (1-t)^{n-i}$ – поліном Бернштейна;

Q_{ij} – елемент i -го рядку j -го стовпця матриці контрольних точок;

n і m – відповідно, кількість рядків і стовпців у матриці контрольних точок.

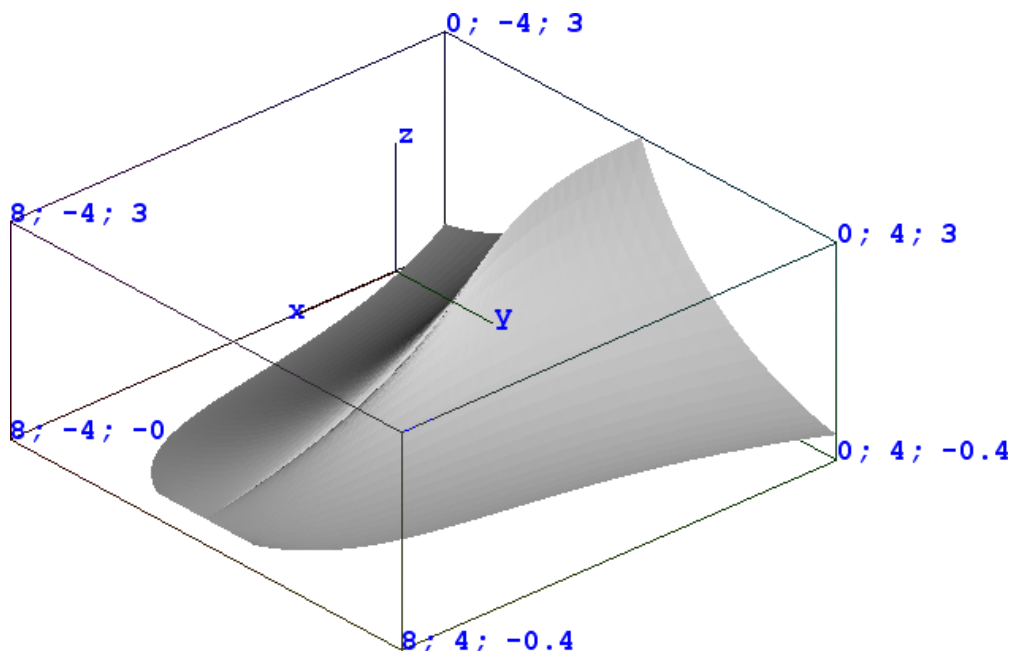


Рисунок 1.48 – Поверхня літального апарату

Розподіли значень функції відстані зі знаком для моделі літального апарату в перетинах площинами $y=0$ і $z=0$, відповідно, наведено на рис. 1.49 і 1.50.

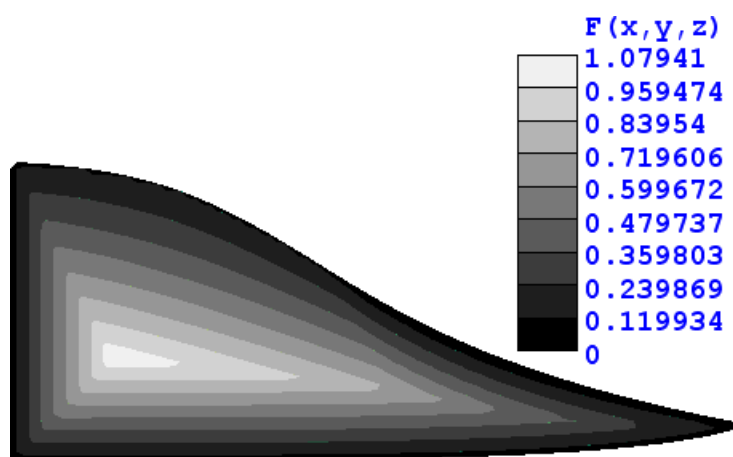


Рисунок 1.49 – Розподіл значень функції відстані зі знаком у перерізі площиною $y=0$

Аналогічно до двовимірного випадку, тривимірні функції відстані можуть виступати як операнди R-операцій для визначення більш складних об'єктів.

Наприклад, для подання моделі літального апарату у вигляді, наведеному на рис. 1.51, можна використати таку формулу:

$$\begin{aligned}
 F(x, y, z) &= f(x, y, z, \text{surface}(\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}, \mathbf{E}, \mathbf{F})) \wedge \\
 &\quad \wedge \text{Torp}(x; y-1,5; z) \wedge \text{Torp}(x; y+1,5; z), \\
 \text{Torp}(x, y, z) &= x \wedge (r^2(x) - y^2 - z^2) \wedge (4-x), \\
 r(x) &= \begin{cases} 1-0,5x, & x \leq 1, \\ 0,5, & 1 < x \leq 3, \\ 0,5-0,5(x-3), & 3 < x. \end{cases}
 \end{aligned}
 \tag{1.51}$$

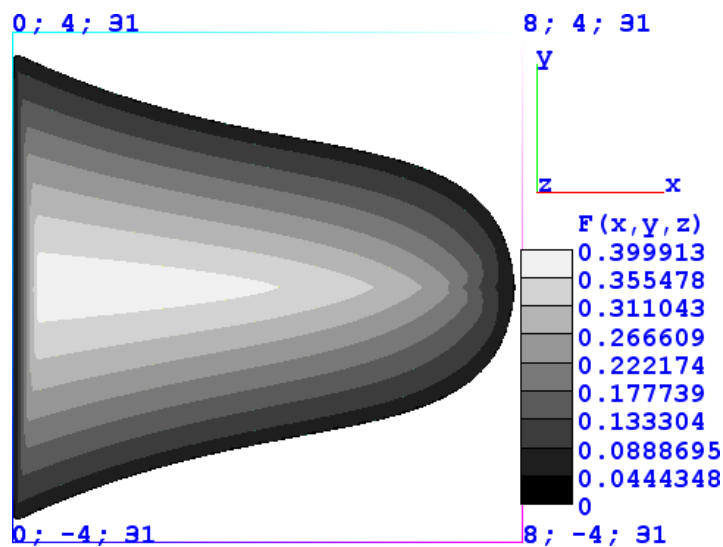


Рисунок 1.50 – Розподіл значень функції відстані зі знаком у перерізі площиною $z=0$

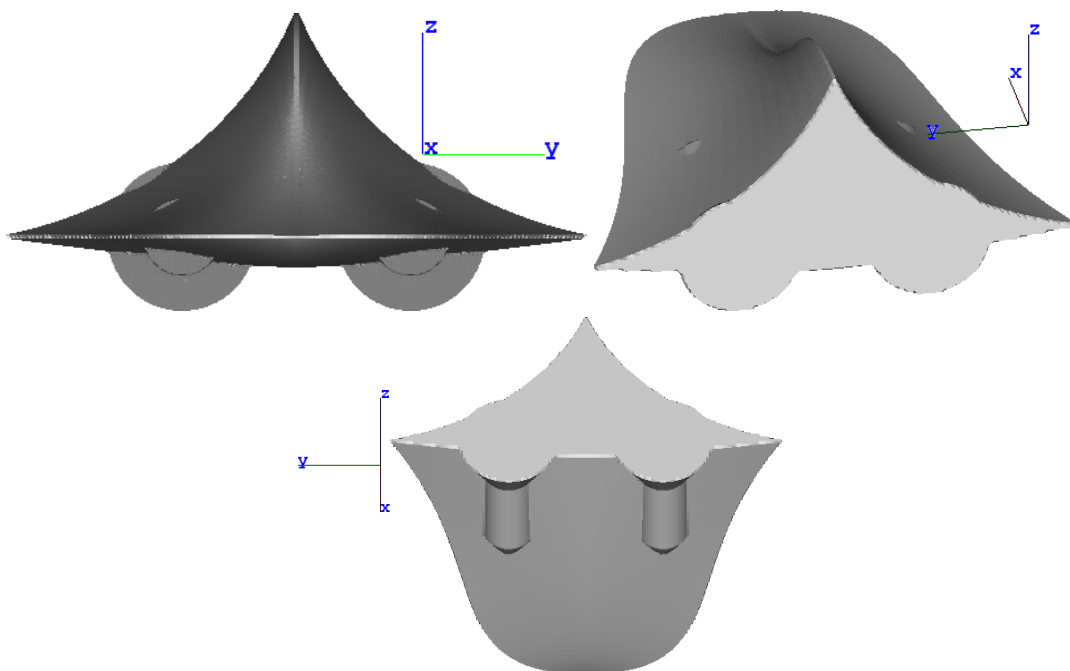


Рисунок 1.51 – Модель літального апарату

Розподіл значень функції (1.51) в перерізі площиною $z=0$ зображено на рис. 1.52.

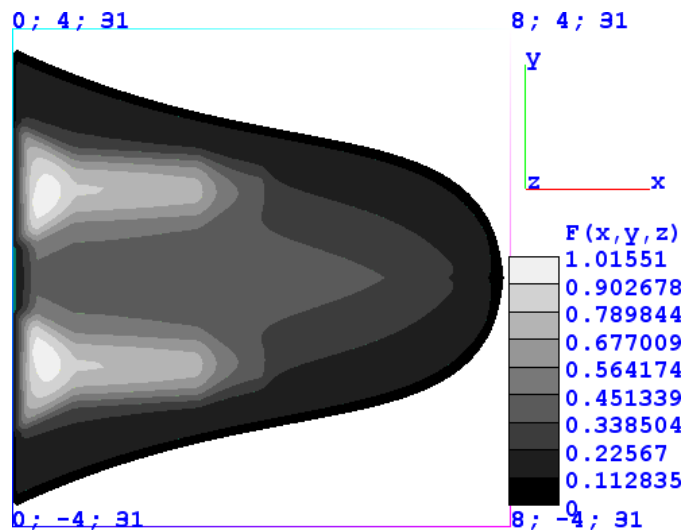


Рисунок 1.52 – Розподіл значень функції (1.51) у перерізі площиною $z=0$

Альтернативно до формули (1.31) поверхню метричної різьби можна подати як результат руху правильного трикутника по спіралеподібній кривій (подібні поверхні в літературі також відомі як поверхні замітання [286]). Сторона трикутника дорівнює кроку різьблення. Кількість витків визначається відношенням висоти робочої поверхні (гайки або болта) до кроку різьби.

Нехай твірна крива – спіраль, радіус якої дорівнює внутрішньому радіусу різьби. Така крива подається формулою її радіус-вектора:

$$g(v) = \mathbf{i} r_i \cos\left(\frac{2\pi v}{m}\right) + \mathbf{j} r_i \sin\left(\frac{2\pi v}{m}\right) + \mathbf{k} v, \quad 0 \leq v \leq L,$$

де $\{\mathbf{i}, \mathbf{j}, \mathbf{k}\}$ – ортонормований базис декартової системи координат тривимірного простору;

r_i – внутрішній радіус різьби;

m – крок різьби;

L – висота робочої поверхні.

Радіус-вектор твірної (правильний трикутник) подається формулою

$$p(u) = \begin{cases} -\frac{m}{2}(1-u)\mathbf{i} + \left(t m \sin\left(\frac{\pi}{3}\right) - \delta\right)\mathbf{j}, & 0 \leq u \leq 1, \\ \frac{m}{2}(u-1)\mathbf{i} + \left[(2-u)m \sin\left(\frac{\pi}{3}\right) - \delta\right]\mathbf{j}, & 1 < u \leq 2, \\ \frac{m}{2}(5-2u)\mathbf{i} - \delta\mathbf{j}, & 2 < u \leq 3, \end{cases}$$

де $\delta = \frac{m \sin\left(\frac{\pi}{3}\right) - (r - r_i)}{2}$ – зсув трикутника;

r – зовнішній радіус різьби.

Радіус-вектор поверхні замітання, яка подає метричну різьбу, задається формулою

$$\text{surf}(u, v) = g(v) + m(v)(p(u) \times (\mathbf{A}(v))^T), \quad (1.52)$$

$$0 \leq u \leq 3, \quad 0 \leq v \leq L,$$

$$\text{де } m(v) = \begin{cases} 5 \frac{v}{L}, & v \leq \frac{L}{5}, \\ 1, & \frac{L}{5} < v < \frac{4}{5}L, \\ 5 \left(1 - \frac{v}{L}\right), & \frac{4}{5}L \leq v, \end{cases} \quad - \text{ залежність для масштабного коефіцієнта, що}$$

забезпечує замкненість поверхні різьби (рис. 1.53);

$$m(v) = \begin{cases} 5 \frac{v}{L}, & v \leq \frac{L}{5}, \\ 1, & \frac{L}{5} < v < \frac{4}{5}L, \\ 5 \left(1 - \frac{v}{L}\right), & \frac{4}{5}L \leq v, \end{cases} \quad - \text{ матриця повороту локальної системи координат}$$

під час руху за траєкторією $g(v)$;

$$\mathbf{A}(v) = \begin{bmatrix} r_i \cos(2\pi v) & r_i \sin(2\pi v) & 0 \\ -\frac{2r_i\pi}{\sqrt{4\pi^2 r_i^2 + 1}} \sin(2\pi v) & \frac{2r_i\pi}{\sqrt{4\pi^2 r_i^2 + 1}} \cos(2\pi v) & \frac{1}{\sqrt{4\pi^2 r_i^2 + 1}} \\ \frac{1}{\sqrt{4\pi^2 r_i^2 + 1}} \sin(2\pi v) & -\frac{1}{\sqrt{4\pi^2 r_i^2 + 1}} \cos(2\pi v) & \frac{2r_i\pi}{\sqrt{4\pi^2 r_i^2 + 1}} \end{bmatrix}.$$

У результаті функція, що подає різьбу, у формулі (1.29), отримає вигляд

$$\text{thread}(x, y, z) = f(x, y, z, \text{surf}(u, v)).$$

Загальний вигляд отриманої моделі і розподіл значень функції (1.29) при параметричному поданні поверхні різьби показані на рис. 1.54. Можна побачити, що підсумкова поверхня має той самий вигляд, але змінився розподіл значень функції.

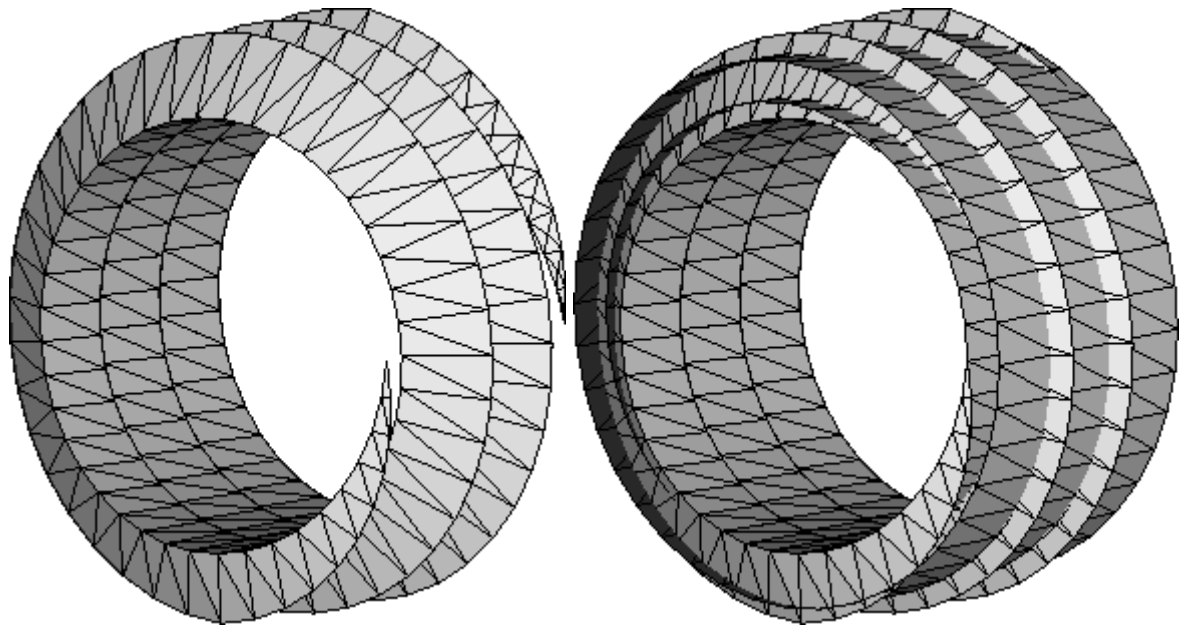


Рисунок 1.53 – Дискретна модель параметричної поверхні різьби

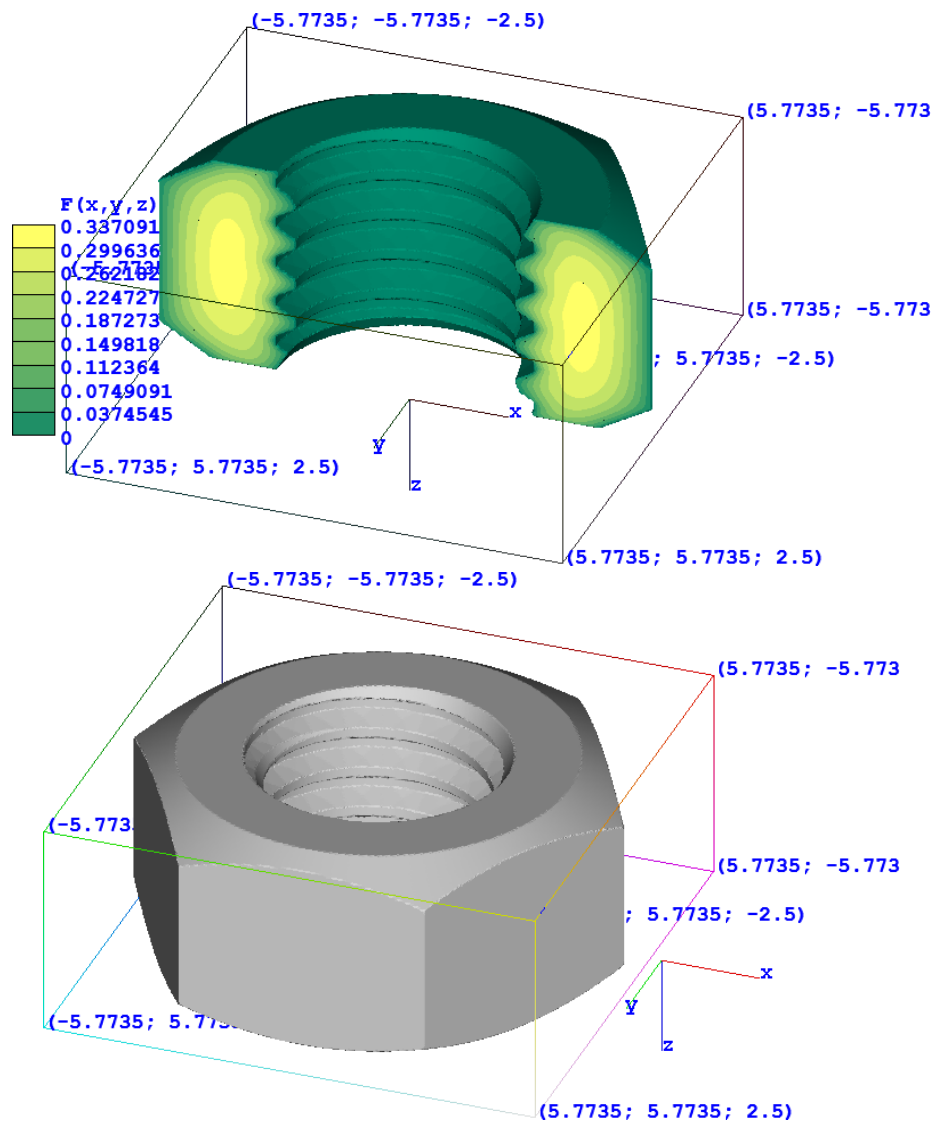


Рисунок 1.54 – Розподіл значень функції (1.29) при гібридному поданні гайки

1.8. Подання оболонкових об'єктів і конструкцій

Автоматизоване проектування багатьох об'єктів аерокосмічної, автомобіле- і суднобудівної галузей пов'язано з моделюванням тривимірних тіл, у яких один з розмірів (товщина) значно менше двох інших. Такі тіла для економії обчислювальних ресурсів модельно подають оболонками у формі параметричних поверхонь вигляду (1.47). Водночас подання оболонкових об'єктів, що володіють вирізами або технологічними отворами різної форми, за допомогою клаптів (параметричних поверхонь, що задають ділянки межі) не завжди зручне. У таких випадках модель оболонкового об'єкта можна розглядати у такому вигляді:

$$\Omega = \{(u, v) \in P \mid F(\text{surface}(u, v)) \geq 0\}, \quad (1.53)$$

де $\text{surface}(u, v)$ – параметрична поверхня, визначена на прямокутнику $P = \{(u, v) \mid a \leq u \leq b, c \leq v \leq d\}$;

$F(x, y, z)$ – неявна функція, яка від'ємна в точках, які не належать об'єкту (параметрична форма функції $\text{surface}(u, v)$ дозволяє явно перерахувати всі точки, що належать об'єкту, тому функцію $F(x, y, z)$ досить побудувати у вигляді, що дозволяє виключити точки, що потрапили в вирізи або отвори).

Наприклад, оболонку, яка задана поверхнею Безьє з отвором (рис. 1.55), отриманим логічним відніманням циліндра з направляючою уздовж осі абсцис і радіусом 0,4, можна подати такою парою функцій:

$$\begin{cases} \text{surface}(u, v) = \text{sbezier}(u, v), \\ F(x, y, z) = -\text{circle}(y; z; 0,4), \end{cases} \quad (1.54)$$

де sbezier – параметрична функція (1.50) з матрицею контрольних точок

$$Q = \begin{bmatrix} (0; -1; -1) & (0; -1; 1) \\ (4; -1; -1) & (4; -1; 1) \\ (4; 1; -1) & (4; 1; 1) \\ (0; 1; -1) & (0; 1; 1) \end{bmatrix};$$

circle – функція (1.11).

Конічну оболонку зі складною конфігурацією технологічних отворів, висота якої спрямована уздовж осі Oy , можна подати у вигляді такої параметричної функції:

$$\text{cone}(u, v) = \left[r_b + u(r_t - r_b) \right] \cos(2\pi v) \mathbf{i} + l u \mathbf{j} + \left[r_b + u(r_t - r_b) \right] \sin(2\pi v) \mathbf{k},$$

$$0 \leq u \leq 1, 0 \leq v \leq 1,$$

де r_b – радіус нижньої основи;

r_t – радіус верхньої основи;

l – висота.

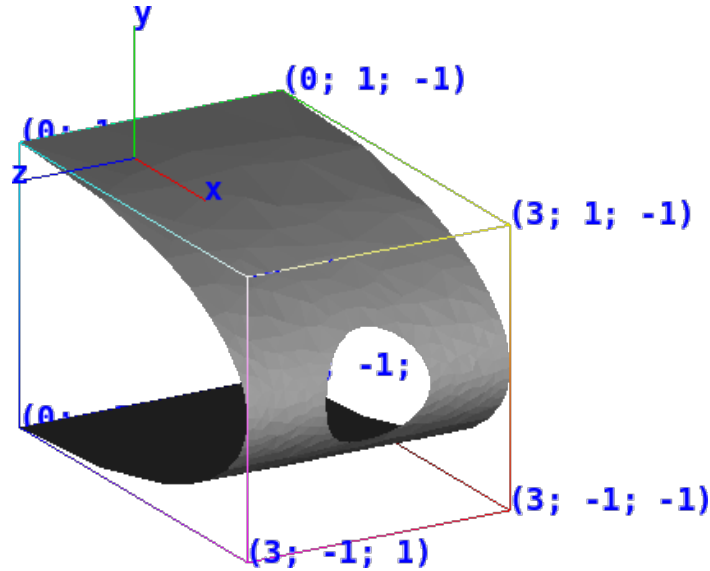


Рисунок 1.55 – Поверхня Безьє з отвором

Технологічні отвори в конічній оболонці у формі прямокутників з округленими кутами (три з яких наскрізні), а також одним круговим отвором можна задати функцією

$$\begin{aligned} F(x, y, z) = & \text{rectangle}_r(\text{rot}_{x,y}(x, z, \alpha_1), y - y_1, w_1, h_1, r_1) \vee \\ & \vee \text{rectangle}_r(\text{rot}_{x,y}(x, z, \alpha_2), y - y_2, w_2, h_2, r_2) \vee \\ & \vee \text{rectangle}_r(\text{rot}_{x,y}(x, z, \alpha_3), y - y_3, w_3, h_3, r_3) \vee \\ & \vee \left[\text{rectangle}_r(\text{rot}_{x,y}(x, z, \alpha_4), y - y_4, w_4, h_4, r_4) \wedge \text{rot}_{z,y}(x, z, \alpha_4) \right] \vee \\ & \vee \left[\text{rectangle}_r(\text{rot}_{x,y}(x, z, \alpha_5), y - y_5, w_5, h_5, r_5) \wedge \text{rot}_{z,y}(x, z, \alpha_5) \right] \vee \\ & \vee \left[\text{rectangle}_r(\text{rot}_{x,y}(x, z, \alpha_6), y - y_6, w_6, h_6, r_6) \wedge \text{rot}_{z,y}(x, z, \alpha_6) \right] \vee \\ & \vee \left[\text{circle}(\text{rot}_{x,y}(x, z, \alpha_7), y - y_7, r_7) \wedge \text{rot}_{z,y}(x, z, \alpha_7) \right], \end{aligned} \quad (1.55)$$

де $\text{rot}_{x,y}(x, z, \alpha) = x \cos(\alpha) + z \sin(\alpha)$;

$\text{rot}_{z,y}(x, z, \alpha) = -x \sin(\alpha) + z \cos(\alpha)$;

rectangle_r – функція (1.16);

circle – функція (1.11).

Наприклад, якщо вважати $\alpha_1 = -0,65$, $\alpha_2 = -1,31$, $\alpha_3 = -2,09$, $\alpha_4 = -1,31$, $\alpha_5 = 2,09$, $\alpha_6 = 0,65$, $\alpha_7 = -2,43$, $w_1 = h_1 = w_2 = h_2 = w_3 = 0,4$, $r_1 = r_2 = 0,055$, $r_3 = 0,065$, $w_4 = w_5 = 0,68$, $h_4 = h_5 = 0,45$, $r_4 = r_5 = r_6 = 0,075$, $w_6 = 0,44$, $h_6 = 0,88$, $y_1 = y_2 = 3,434$, $y_3 = 3,529$, $y_4 = y_5 = 0,52$, $y_6 = 0,7$, $y_7 = 3,554$, а також параметри конічної оболонки $r_b = 1,5$, $r_t = 1,99$ і $l = 4,014$, то отримаємо конфігурацію отворів у вигляді, наведеному на рис. 1.56.

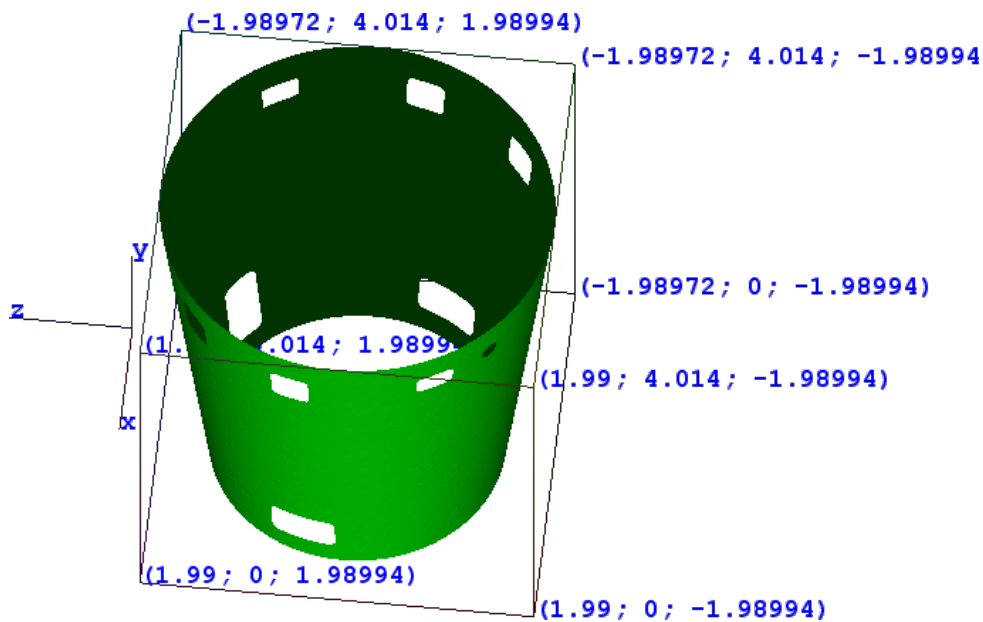


Рисунок 1.56 – Конічна оболонка з отворами

При такому підході процедури, пов'язані з обробкою математичної моделі об'єкта (наприклад, побудову та поліпшення дискретних моделей), можна виконувати у двовимірному параметричному просторі, зменшуючи, таким чином, розмірність задачі.

1.9. Подання дискретних моделей у САПР

Моделі, які ґрунтуються на чисельному розв'язанні рівнянь в частинних похідних, часто використовують в задачах проектування та дослідження технічних об'єктів. Один з найбільш поширених на практиці чисельних методів – метод скінченних елементів – (як і деякі інші) замінює вихідну неперервну модель дискретною, яку можна обробити комп'ютером. У результаті на першому кроці моделювання для деякої неперервної області необхідно побудувати скінченну множину простих фігур (елементів), наприклад, трикутників, чотирикутників, тетраедрів, шестигранників тощо.

Взаємне розташування вершин елементів істотно впливає на точність чисельного аналізу. Отже, автоматизація побудови дискретних моделей геометричних об'єктів є одним з ключових етапів реалізації хмарної САПР машинобудування.

Інтуїтивно, дискретна модель форми об'єкта – це колекція вузлів і елементів, яка визначає форму цього об'єкта. Координати вузлів визначають геометрію об'єкта, елементи – його топологію. У загальному випадку, елементи можуть мати тільки спільні вершини, ребра і грані. За такої умови для кожної пари вузлів, яка є ребром в одному елементі, має існувати ребро в іншому елементі, якщо обидва елементи інцидентні в цих вузлах. Аналогічну вимогу можна сформулювати щодо граней. Ці дві вимоги необхідні для забезпечення неперервності чисельного рішення, відповідно, вздовж ребер і граней.

Формальне визначення дискретної моделі форми об'єкта – це введення відповідної структури даних. Воно необхідне тому, що пара алгоритм і структура даних є основою для створення програми. Водночас розробка загальної структури даних для довільної дискретної моделі є досить складною задачею.

Історично першими отримали поширення полігональні дискретні моделі поверхонь об'єктів. Для них розроблені досить ефективні програмні й апаратні методи візуалізації (наприклад, за допомогою зворотного трасування променів). Багатокутник в порядку обходу однозначно визначається координатами своїх вершин. Відповідно, у задачах візуалізації для кожного полігону досить зберігати тільки координати його вершин. В результаті на кожен полігон буде виділятися $3kf$ байт пам'яті (k – кількість вершин, f – кількість байт, що використовується для зберігання одного числа з рухомою комою).

У задачах генерації й обробки дискретних моделей виникає необхідність вставки або видалення вузлів. У разі зберігання елементів у вигляді масивів координат необхідний пошук елементів зі спільними вузлами. В результаті, в системах інженерного аналізу поширення отримали структури даних, засновані на поданні вершин елементів вказівниками на позиції у масиві або списку координат вершин. Водночас елементи володіють додатковою топологічною інформацією, яка спрощує знаходження сусідів. Піонером у цьому напрямку можна вважати схему «крилате ребро» (winged edge) [287]. У ньому кожен вузол зберігає інформацію про свої координати та посилання на одне зі своїх ребер, грань зберігає посилання на ребра в порядку їх обходу. Ребро зберігає посилання на два вузла та дві грані (умовно ліву та праву), а також чотири «крила»: посилання на попередні та наступні ребра в порядку обходу обох граней. Схожі ідеї закладені у такі схеми: «радіальне ребро» (radial-edge) [288], «орієнтоване ребро» (directed edge) [289], «півребро» (half-edge) [290], а також узагальнення останнього [291–293]. Подібні схеми досить ефективні під час

зберігання та обробки полігональних моделей, але їх дуже проблематично застосовувати для об'ємних елементів.

Компактні схеми збереження об'ємних елементів розроблені для тетраедрів у вигляді узагальнення схем для трикутників [292, 294]. Одночасно, у роботі [295] запропоновано схему стисненого збереження дискретних моделей на основі шестигранників.

Розробка систем інженерного аналізу потребує єдиної абстрактної структури даних, яка не залежить від форми елемента. Її введення дозволить використовувати єдину формальну нотацію для викладання алгоритмів.

1.9.1 Формальне визначення дискретної моделі форми об'єкта машинобудування

Формально, дискретна модель форми об'єкта – це пара множин:

$$\mathbf{M}=(\mathbf{V}, \mathbf{E}), \quad (1.56)$$

де $\mathbf{V}=(v_{k_1}, v_{k_2}, \dots, v_{k_m})$ – множина координат вузлів;

$\mathbf{E}=(e_1=(k_{1,1}, k_{1,2}, \dots, k_{1,n_1}), e_2=(k_{2,1}, k_{2,2}, \dots, k_{2,n_2}), \dots, e_q=(k_{q,1}, k_{q,2}, \dots, k_{q,n_q}))$ – множина елементів, кожен з яких має $n_i = |e_i|$ вершин, – кортежів кодів вузлів з множини $\mathbf{V}$, що визначають положення вершин елемента;

m – кількість вузлів;

q – кількість елементів.

Координати j -го елемента з n вершинами визначаються підстановкою $\mathbf{V}(e_j)=(v_{k_{j,1}}, v_{k_{j,2}}, \dots, v_{k_{j,n}})$.

Множину $\mathbf{V}$, у загальному випадку, можна вважати асоціативним масивом, який зберігає пари ключ-значення. Ключем виступає код вузла (номер, або адреса у пам'яті), значенням – його координати. Відповідно, запис $\mathbf{V}[k]$ позначає координати (радіус-вектор) вузла $v_k \in \mathbf{V}$.

Аналогічне твердження справедливе для множини $\mathbf{E}$, в якій ключами можуть виступати коди (номери або адреси) елементів, а значеннями – відповідні списки або масиви, що зберігають кортежі. У такому випадку запис $\mathbf{E}[i]$ позначає елемент $e_i \in \mathbf{E}$. Враховуючи, що e_i є кортежем, $e_i[j] = \mathbf{E}[i][j]$ позначає код вузла $k_{i,j}$, який визначає положення j -ї вершини.

Для більш короткого позначення операцій щодо дискретної моделі $\mathbf{M}$ у квадратних дужках записані операції доступу до елементів множини $\mathbf{V}$, а у

круглих дужках – до елементів множини $\mathbf{E}$. Тобто, $\mathbf{M}[k] = \mathbf{V}[k]$, $\mathbf{M}(i) = \mathbf{E}[i]$, $\mathbf{M}(i, j) = \mathbf{E}[i][j]$. У такому випадку $|\mathbf{M}(i)|$ – кількість вершин (вузлів) у елементі з кодом i .

1.9.2 Операції вставки та видалення, пошуку сусідів

Алгоритми вставки та видалення вузлів і елементів для дискретної моделі (1.56) можна віднести до елементарних. Додавання нового елементу потребує почергового додавання всіх його вершин. У свою чергу, вставка нового вузла потребує перевірки наявності вузла з такими ж координатами. Результатом операції вставки буде код вузла в множині $\mathbf{V}$ (новий або наявний, якщо такий самий вузол був знайдений). При видаленні вузла також необхідно буде видалити інцидентні елементи, а під час видалення елемента необхідно видалити тільки вузли, які більше не беруть участь у зв'язках.

Для дискретної моделі, що складається з m вузлів і n елементів, кожен з яких має q вершин, операція вставки вузла може бути реалізована за допомогою лінійного алгоритму з обчислювальною складністю $O(m)$. Додавання елемента, заданого координатами власних вершин, q раз вимагатиме виконання вставки вузла та, відповідно, буде мати обчислювальну складність $O(qm)$. Для формування множини інцидентних деякому вузлу елементів досить буде одного проходу по елементам множини $\mathbf{E}$, отже, обчислювальна складність цієї процедури буде дорівнювати n операцій. У результаті під час видалення одного елемента буде потрібно nq операцій, а під час видалення вузла – $(kn+1)q$ операцій (k – кількість інцидентних у вузлі елементів).

Очевидним недоліком дискретної моделі форми об'єкта (1.56) є необхідність використання лінійного пошуку для формування списків інцидентних елементів. Ці операції виникають під час видалення вузлів та елементів, а також поширені у обробці дискретних моделей (наприклад, локальному згладжуванні). Для усунення цього недоліку можна припустити, що кожен вузол крім свої координат зберігає ще інформацію про множину суміжних елементів. У результаті дискретну модель форми об'єкта можна подати у такому вигляді:

$$\mathbf{M} = (\mathbf{N}, \mathbf{E}), \quad (1.57)$$

де $\mathbf{N} = ((v_{k_1}, \mathbf{I}_{k_1}), (v_{k_2}, \mathbf{I}_{k_2}), \dots, (v_{k_m}, \mathbf{I}_{k_m}))$ – множина пар: v_{k_i} – координати вузла з кодом k_i та $\mathbf{I}_{k_i}$ – коди інцидентних йому елементів;

E – таке ж, як у (1.56).

Для дискретної моделі форми об'єкта (1.57) операції доступу можна визначити у такому вигляді: $M[k] = N[k] = v_k$, $M(i) = E[i]$, $M(i, j) = E[i][j]$. До цих операцій можна додати такі: $M.incident(k) = I_k$ – отримання множини кодів елементів, інцидентних у вузлі з кодом k , і $M.code(n)$ – визначення коду параметра (вузла або елемента). Також припустимо, що для кожного $n \in N$ запис $n.v$ відповідає координатам, $n.I$ – множині інцидентних йому елементів.

Вставка вузла у дискретну модель (1.57) аналогічна описаній вище, але вимагає ініціалізації множин кодів інцидентних елементів порожніми. У результаті вставку вузла можна формалізувати таким алгоритмом:

algorithm insert_node

input:

M – дискретна модель форми об'єкта

p – координати вузла

ε – чисельний ноль

output:

k – код вузла

begin

for each $n \in M.N$ **do**

if $\|p - n.v\| < \varepsilon$ **then**

return $M.code(n)$

end if

end for

$M.N \leftarrow M.N \cup \{(p, \emptyset)\}$

return $M.code((p, \emptyset))$

Алгоритм вставки нового елемента, заданого списком координат власних вершин, в дискретну модель в обох випадках можна записати у такому вигляді:

algorithm insert_element

input:

M – дискретна модель форми об'єкта

V – список координат вершин

ε – чисельний нуль

begin

$(e, k_e) \leftarrow$ Новий елемент з $|V|$ вершинами та його код у $M.E$

for $i \leftarrow 1$ **to** $|V|$ **do**

$e[i] \leftarrow insert\_node(M, V[i], \varepsilon)$

$M.incident(e[i]) \leftarrow M.incident(e[i]) \cup \{k_e\}$

end for

$M.E \leftarrow M.E \cup \{e\}$

end

Операція видалення елемента з дискретної моделі (1.57) потребує оновлення множин інцидентних елементів у вузлах, відповідних вершин. У результаті, цю операцію можна описати у вигляді такого алгоритму:

algorithm remove_element

input:

$\mathbf{M}$ – дискретна модель форми об'єкта

k_e – код елемента

begin

$e \leftarrow \mathbf{M}(k_e)$

for $i \leftarrow 1$ **to** $|e|$ **do**

$\mathbf{M}.\text{incident}(e[i]) \leftarrow \mathbf{M}.\text{incident}(e[i]) \setminus \{k_e\}$

if $\mathbf{M}.\text{incident}(e[i]) = \emptyset$ **then**

 remove_node($\mathbf{M}$, $e[i]$)

end if

end for

$\mathbf{M}.\mathbf{E} \leftarrow \mathbf{M}.\mathbf{E} \setminus \{e\}$

end

Операцію видалення вузла, яка використовується вище, можна реалізувати у вигляді такого алгоритму:

algorithm remove_node

input:

$\mathbf{M}$ – дискретна модель форми об'єкта

k_n – код вузла

begin

$\mathbf{I} \leftarrow \mathbf{M}.\text{incident}(k_n)$

for each $k_e \in \mathbf{I}$ **do**

 remove_element($\mathbf{M}$, k_e)

end for

$\mathbf{M}.\mathbf{N} \leftarrow \mathbf{M}.\mathbf{N} \setminus \{\mathbf{M}.\mathbf{N}[k_n]\}$

end

Операції видалення можна вважати тісно зв'язними між собою, тому що, реалізація однієї з них залежить від реалізації іншої, що підкреслює важливість акуратності їх програмної реалізації.

Операція пошуку сусідніх вузлів ґрунтується на аналізі інцидентних елементів і для усунення неоднозначності потребує введення обмежень на порядок обходу вершин у визначенні елемента. Припустимо, що у двовимірних

елементів порядок обходу вершин проти годинникової стрілки, а у тривимірних – у порядку обходу проти годинникової стрілки спочатку нижніх граней, а потім верхніх (з урахуванням обертання, де це може бути застосовано, рис. 1.57).

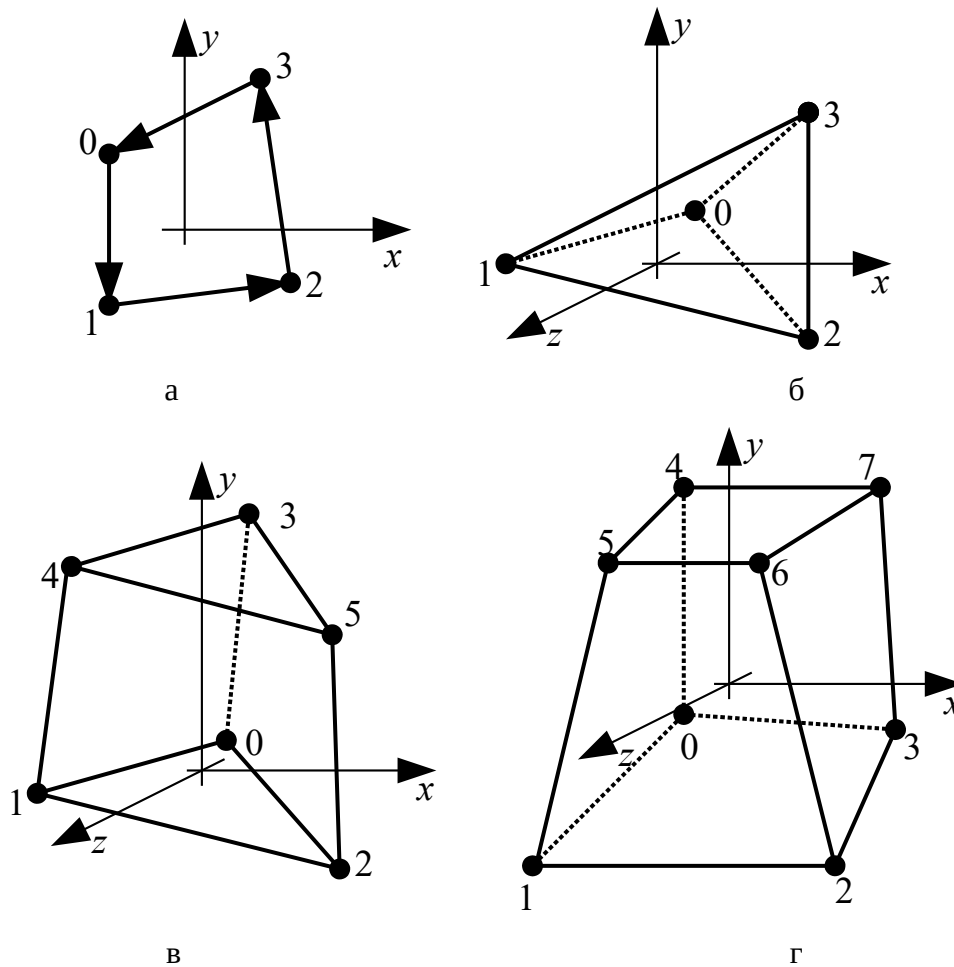


Рисунок 1.57 – Порядок обходу в елементах: а – порядок обходу для двовимірних елементів; б – порядок обходу для тетраедрів; в – порядок обходу для трикутних призм; г – порядок обходу для шестигранників

Для забезпечення можливості циклічного обходу вузлів одного елемента припустимо, що операція доступу до коду вершини за її номером визначена у такому вигляді:

$$\mathbf{M}(i, j) = \mathbf{M} \cdot \mathbf{E}[i][j] = \begin{cases} j \bmod |e_j|, & j \geq 0 \\ 0, & j < 0 \wedge (j \bmod |e_j|) = 0, \\ |e_j| + (j \bmod |e_j|), & j < 0 \wedge (j \bmod |e_j|) \neq 0. \end{cases}$$

У результаті множина номерів (кодів) вузлів, сусідніх з j -ю вершиною i -го елемента, можна визначити у наступному вигляді:

$$\text{neighbours}(\mathbf{M}, i, j) = \begin{cases} \{\mathbf{M}(i, j-1), \mathbf{M}(i, j+1)\}, & \text{для плоских елементів,} \\ \{\mathbf{M}(i, j-1), \mathbf{M}(i, j+1), \mathbf{M}(i, j+f)\}, & \text{для об'ємних елементів,} \end{cases}$$

де f кількість вузлів у нижній грані ($f = 3$ у тетраедрів і трикутних призм, $f = 4$ у шестигранників).

Для формування множини номерів (кодів) вузлів, суміжних у вузлі з номером k_n , необхідно об'єднати множини номерів вузлів, сусідніх з цією вершиною всіх інцидентних у k_n елементів. Цю процедуру можна подати таким алгоритмом.

algorithm adjacent

input:

$\mathbf{M}$ – дискретна модель форми об'єкта

k_n – код вузла

output:

A – множина номерів (кодів) вузлів, суміжних у вузлі з номером k_n

begin

$A \leftarrow \emptyset$

$I \leftarrow \mathbf{M}.\text{incident}(k_n)$

for each $k_e \in I$ **do**

$e \leftarrow \mathbf{M}(k_e)$

$j \leftarrow (j \in 0, \dots, |e|-1 : \mathbf{M}(k_e, j) = k_n)$ // j – номер вершини у e з кодом k_n

$A \leftarrow A \cup \text{neighbors}(\mathbf{M}, k_e, j)$

end for

return A

Алгоритм `adjacent` є лінійним і потребує тільки обробки множини інцидентних елементів, яка, зазвичай, є невеликою. Його обчислювальна складність для вузла з номером k_n є величиною порядку $O(|\mathbf{M}.\text{incident}(k_n)|)$ операцій. У подальшому викладі побудова множини кодів суміжних вузлів для вузла з кодом k_n позначена $\mathbf{M}.\text{adjacent}(k_n)$.

1.9.3 Структури даних

Зберігання дискретної моделі форми об'єкта (1.57) можливе декількома способами. Ключові відмінності між ними знаходяться у типах даних, які використовуються для зберігання вершин та елементів. Традиційно, коли мова йде про множини однотипних екземплярів, протиставляють дві структури даних: векторну і спискову.

Векторна структура даних базується на розміщенні екземплярів у послідовних адресах пам'яті комп'ютера, що дозволяє здійснювати доступ до елемента за його індексом за $O(1)$ часу. За такої умови видалення або вставка у середину нових екземплярів для такої структури даних пов'язані з лінійними витратами часу, викликаними зсувами масивів даних.

Спискова структура даних ґрунтується на включенні додатково до інформаційних полів адресних полів-показчиків, які служать для визначення зв'язків у списку. Вони дозволяють швидко видаляти та вставляти примірники у список, але призводять до збільшення витрат пам'яті комп'ютера. Також їх ітерування, зазвичай, є більш повільним за рахунок більшої фрагментації оперативної пам'яті.

Розглянемо дві можливі реалізації: 1 – що ґрунтується на використанні динамічних масивів; 2 – що ґрунтується на використанні двозв'язних списків для зберігання вузлів та елементів. Для дослідження продуктивності використаємо модельні задачі: 1 – додавання довільного набору шестигранних елементів, заданих координатами вершин; 2 – видалення довільного набору вузлів.

Як тестовий використовується комп'ютер з процесором Intel Core i5, 8 гігабайтами оперативної пам'яті, який знаходиться під управлінням операційної системи на базі Linux. Кожен тест запускається тричі, у результаті фіксується найменший час, витрачений на виконання завдання.

На рис. 1.58 показана залежність часу побудови дискретної моделі від кількості елементів. На осі абсцис основне значення показує кількість елементів у моделі, а значення у дужках – вузлів.

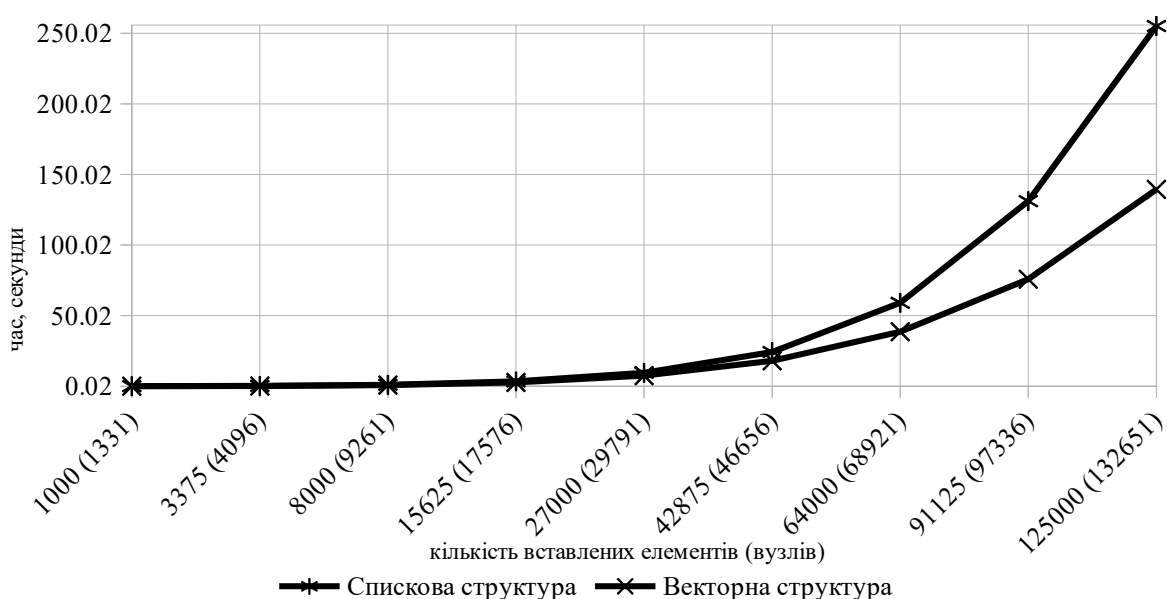


Рисунок 1.58 – Залежність часу побудови моделі від кількості елементів і вузлів

Побудова дискретної моделі, яка ґрунтується на динамічних масивах (векторна структура), потребує меншого часу відносно аналогічної, заснованої на списковій структурі. В обох випадках графіки мають вигляд степеневих функцій. Вигляд графіка обумовлюється тим, що обчислювальну складність вставки m вершин з перевіркою існування можна оцінити величиною $O\left(\frac{m(m-1)}{2}\right)$.

На рис. 1.59 показана залежність часу видалення тисячі вузлів від загальної їх кількості у дискретній моделі форми об'єкта. Необхідність лінійних зсувів обумовлює істотно більш повільне видалення (близько 8 разів) при використанні векторних структур для зберігання вузлів і елементів у порівнянні зі списковими структурами. У обох випадках залежність часу від кількості вузлів у моделі близька до лінійної.

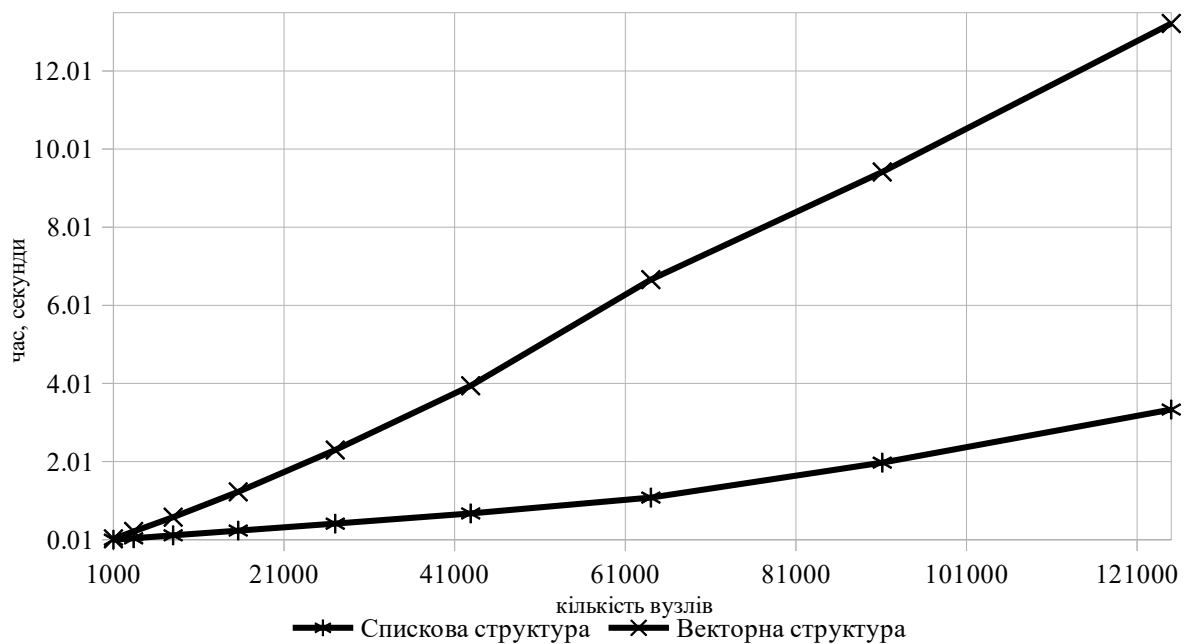


Рисунок 1.59 – Залежність часу видалення 1000 вузлів від кількості вузлів у моделі

Для дослідження впливу кількості видалених вузлів на час алгоритму роботи розглянемо дискретну модель (рис. 1.60), яка складається з 27000 вузлів і 24389 елементів. Як у попередньому випадку, видалення вузлів при використанні спискових структур для зберігання вузлів і елементів істотно швидше векторних. Необхідно відзначити, що операції видалення вузлів і елементів при побудові дискретних моделей використовуються рідше за вставку та ітерування вузлів, в яких більш ефективними себе показали векторні структури даних.

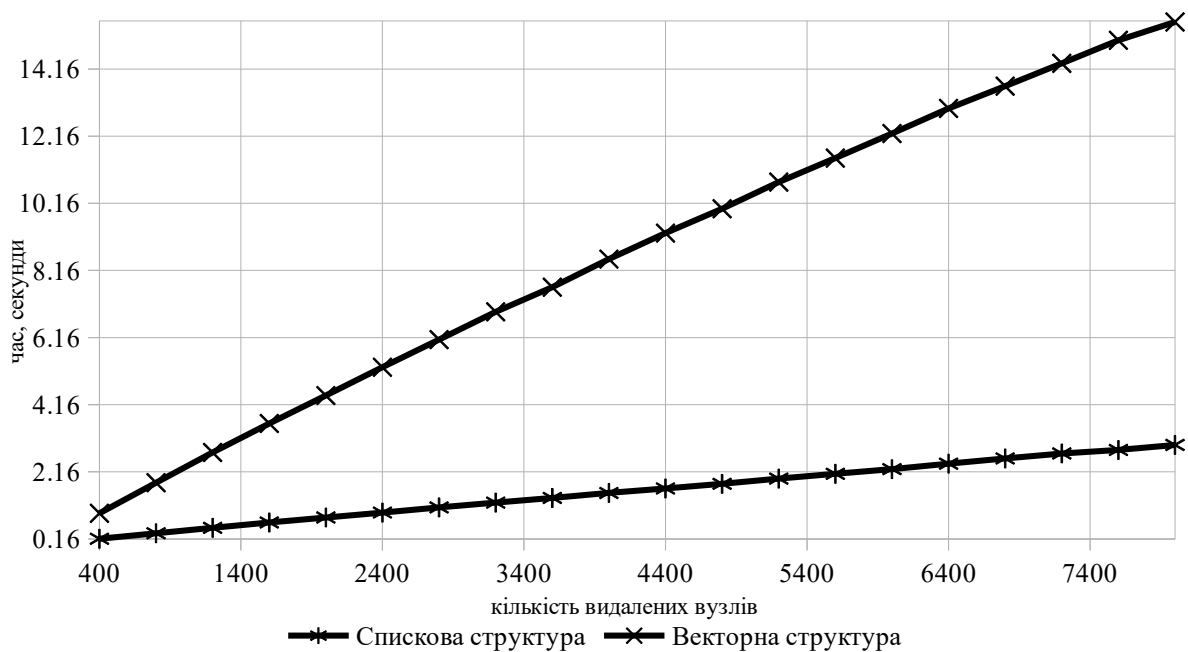


Рисунок 1.60 – Залежність часу від кількості видалених вузлів

1.10. Дискретні моделі меж об'єктів машинобудування

Задачі побудови дискретних моделей меж двовимірних або тривимірних тіл, найчастіше, виникають під час розробки засобів їх візуалізації, тривимірного друку, а також чисельного аналізу на базі методу граничних елементів. Водночас основною вимогою, що висуваються до таких дискретних моделей (у додатках до чисельного аналізу), є точність апроксимації меж. Характерні точки та лінії (точки та лінії зламу поверхні) повинні бути подані вузлами та ребрами елементів дискретних моделей. Щільність елементів повинна бути адаптивною до кривини меж об'єкта.

Одним з перших та водночас найбільш ефективних (з точки зору швидкості роботи) алгоритмів побудови дискретних моделей меж форм об'єктів, поданих функціонально, є метод крокуючих кубиків (marching cubes) [81]. Цей метод заснований на ідеї сканування тривимірної області кубиками зі сталим або адаптивним кроком. У вершинах кожного кубика обчислюють значення функції, що подає форму об'єкта. Потім отримані значення використовують при визначенні відповідного шаблону для тріангуляції ділянки межі, пересіченої ребрами кубика. Всього таких шаблонів 256 (2^8), проте, унікальних (з точністю до обертання) з них тільки 16 (рис. 1.61, чорним кольором позначені внутрішні вершини кубиків). Для швидкого визначення ребер, які пересічені межею, та відповідного шаблону розроблені спеціальні таблиці. Кожному шаблону зіставлений двійковий код, що відповідає його

позиції у цих таблицях. У оригінальному алгоритмі точки перетину ребер межами [81] запропоновано знаходити лінійною інтерполяцією.

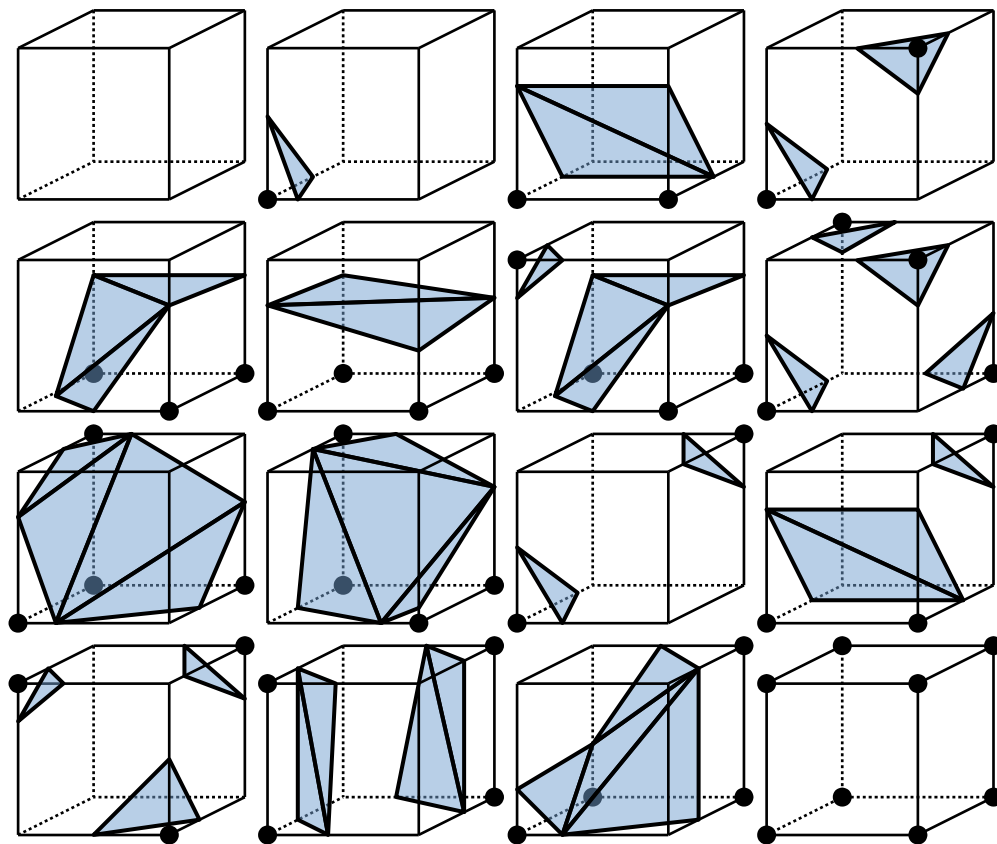


Рисунок 1.61 – Шаблони тріангуляції

Природним аналогом крокуючих кубів на площині є метод крокуючих квадратів (marching squares), який використовує квадрати для сканування площини [296]. У двовимірному випадку шаблонів перетину квадратів межею області буде 16 (2^4), з яких унікальними (з точністю до обертання) можна вважати 6 (рис. 1.62).

Програмні реалізації обох методів, зазвичай, засновані на присвоєння кожному шаблону двійкового коду, який визначається для кожного осередку зміною біта, що відповідає номеру вузла у порядку обходу. Код визначає номери рядків у двох таблицях: таблиці сторін (номерів пар вершин), пересічених границею, та таблиці межових елементів (пар або трійок номерів сторін, на яких знайдені перетини з границею).

Основним недоліком цих методів є можливість отримання незв'язних дискретних моделей у результаті застосування неоднозначних шаблонів. Наприклад, двовимірних шаблонів з кодами 5 та 10 (рис. 1.63) або тривимірних шаблонів, у яких є грані з двома внутрішніми вершинами на одній діагоналі та двома зовнішніми на іншій.

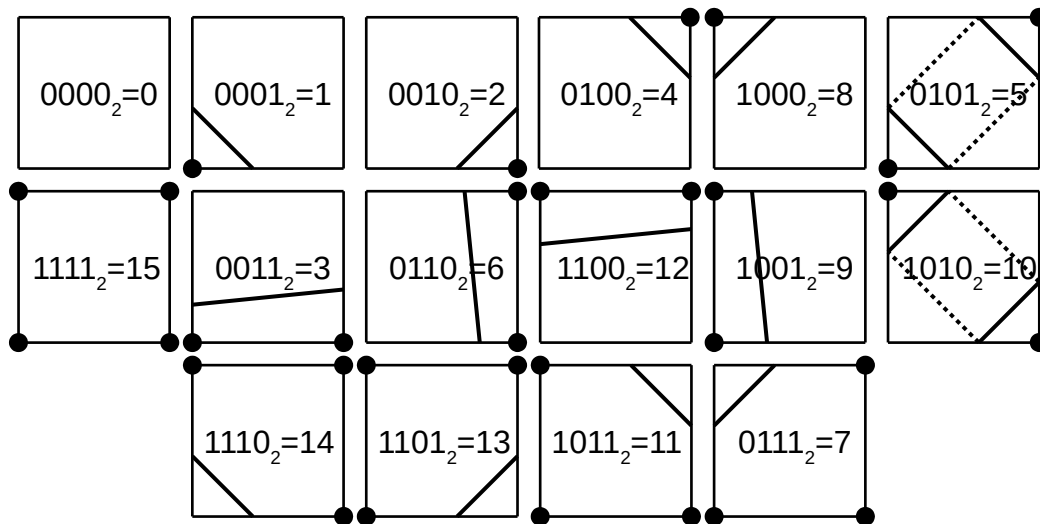


Рисунок 1.62 – Шаблони метода крокуючих квадратів

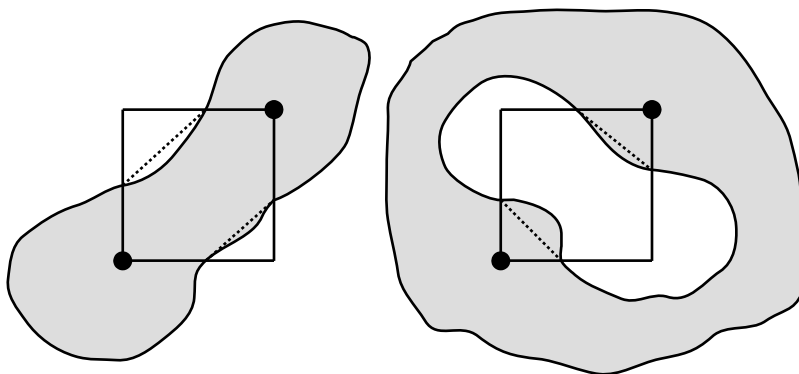


Рисунок 1.63 – Неоднозначність двовимірного шаблону з кодом 5

Найпростішим способом вирішення проблеми незв'язності дискретних моделей, отриманих методом крокуючих квадратів, є рекурсивне розбиття неоднозначних осередків на чотири рівні частини. Водночас для отримання топологічно коректних моделей у тривимірному випадку необхідно виконати додатковий аналіз. У роботі [297] запропоновано використовувати білінійну інтерполяцію для усунення неоднозначності випадків, подібних двовимірним з кодами 5 та 10, для граней кубів. У роботі Черняєва Є. В. [298] підхід, який ґрунтується на білінійній інтерполяції, узагальнено та виділено 33 основних шаблони, а також критерії їх використання (даний метод відомий під назвою «Marching Cubes 33»). Особливості програмної реалізації методу Marching Cubes 33 розглянуті у [299, 300].

Побудова дискретних моделей меж тривимірних форм об'єктів з використанням елементів, форма яких відмінна від трикутників (наприклад, чотирикутників), пов'язана з використанням більш складних алгоритмів. Використання багатьох бібліотек підпрограм для побудови дискретних моделей форм об'єктів виконується за принципом чорного ящика. У той же час, розробка складних програмних продуктів вимагає забезпечення максимальної універсальності рішень, у тому числі, за рахунок залучення патернів (шаблонів)

проектування [301–304]. Можливим способом підвищення технологічності та універсальності програмного коду САПР машинобудування є розробка узагальнених методів побудови дискретних моделей форм об'єктів.

1.10.1 Проекційний метод генерації дискретних моделей меж об'єктів, форма яких задана функціонально

Нехай форма деякого об'єкта Ω подана функцією $F(P)$, яка визначена у кожній точці P евклідового простору. Причому $F(P) > 0$, якщо $P \in \Omega \setminus \delta(\Omega)$, $F(P) = 0$, якщо $P \in \delta(\Omega)$ і, відповідно, $F(P) < 0$, якщо $P \notin \Omega$. Задача полягає у розробці метода побудови дискретної моделі межі об'єкта Ω на базі довільної форми скінченного елемента.

Методи суперпозиції, які було запропоновано у [212–214] для побудови сіток чотирикутників (у двовимірному випадку) та шестигранників (у тривимірному випадку), можна узагальнити на випадок довільної розмірності та форми елемента. У основі цих методів лежить ідея використання деякої базової стратегії адаптації початкової сітки до меж форми деякого об'єкта.

1.10.1.1 Загальний алгоритм метода

Припустимо, що для частини простору, яка повністю включає об'єкт Ω , задана деяка фонові сітка. Якщо видалити з фонові сітки комірки, зовнішні відносно Ω або які перетинаються його межею, отримаємо початкову сітку – множини елементів, яку можна вважати образом цього об'єкта. Потім для кожної межевої грані початкової сітки можна знайти відповідну проекцію на межі об'єкта. У результаті отримаємо такий універсальний алгоритм:

algorithm background_grid_surface

input:

F – функція, яка подає форму об'єкта Ω

G – фонові сітка

ε – чисельний нуль

output:

M – дискретна модель $\delta(\Omega)$

begin

for each $n \in G.N$ **do**

```

if  $F(n.v) < 0$  then
    remove_node( $G, G.code(n)$ )
end if
end for
 $V_0 \leftarrow$  вузли дотику елементів  $G$ 
while  $V_0 \neq \emptyset$  do
     $T \leftarrow \bigcup_{v \in V_0} G.incident(v)$ 
     $k_{e0} \leftarrow \operatorname{argmin}_{k_e \in T} (G.power(G(k_e)))$ 
    remove_element( $G, k_{e0}$ )
     $V_0 \leftarrow$  вузли дотику елементів  $G$ 
end while
 $M \leftarrow \delta(G)$ 
 $h \leftarrow \min_{(n_0 \neq n_1) \in M.N} \|n_0.v - n_1.v\|$ 
for each  $n \in M.N$  do
     $n.v \leftarrow \text{find\_border}(n.v, F, h, \varepsilon)$ 
end for
return  $M$ 

```

Отже, у алгоритмі можна виділити три основні етапи. Першим етапом є видалення всіх зовнішніх вузлів та інцидентних їм елементів фонові сітки. У результаті залишаються вузли та елементи, які можна вважати початковим дискретним образом форми об'єкта Ω .

На другому етапі усуваються конфігурації, які можуть спричинити топологічно невірні результати. Такі конфігурації утворюються у разі дотику елементів у вузлі (приклад такої конфігурації для двовимірного випадку зображений на рис. 1.64). У даному випадку елементи вважаються сусідніми у вузлі, якщо у його оточенні простір ділиться на дві зв'язних підмножини, інакше вони вважаються дотичними.

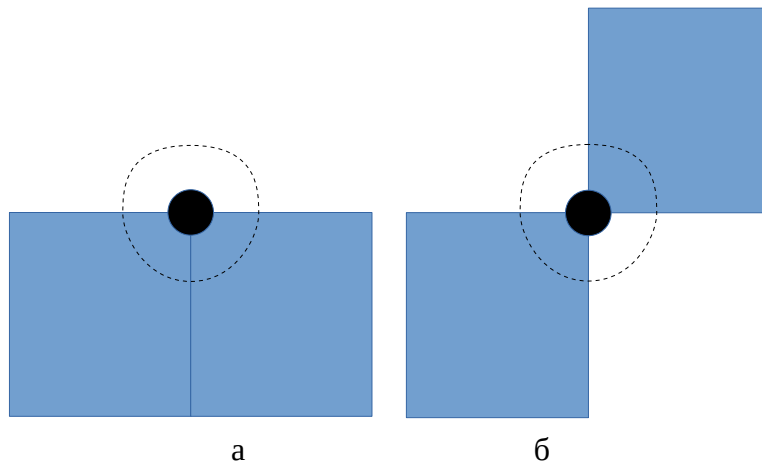


Рисунок 1.64 – Взаємне розташування елементів: а – сусідні елементи; б – дотичні елементи

На третьому етапі формується підсумкова дискретна модель. Спочатку формується множина межових граней для елементів, які залишилися після другого етапу (ця дія позначена оператором δ). Потім для кожного межового вузла знаходиться проекція на межі об'єкта Ω .

1.10.1.2 Алгоритм пошуку проекцій вузлів на межу об'єкта

Для проектування вузла на межу використовується алгоритм `find_border(v, F, h,  $\epsilon$ )`. Він знаходить проекції точки v на границю функції F з кроком h у напрямку градієнта функції F (ϵ – чисельний нуль). Крок h визначається в алгоритмі `background_grid_surface` як мінімальна відстань між парою сусідніх вузлів. Формально алгоритм `find_border` можна записати у такому вигляді:

algorithm `find_border`

input:

v – координати точки, для якої необхідно знайти проекцію на межу об'єкта Ω

F – функція, що подає форму об'єкта Ω

h – крок метода

ϵ – чисельний нуль

output:

точка на межі Ω

begin

$f \leftarrow F(v)$

while $|f| \geq \epsilon$ **do**

$p \leftarrow (v - \text{sign}(f) \cdot h \cdot \text{grad } F(v))$

if $\text{sign}(f) \neq \text{sign}(F(p))$ **then**

return `binary(v, p, F,  $\epsilon$ )`

end if

$v \leftarrow p$

$f \leftarrow F(v)$

end while

return v

Алгоритм `find_border` здійснює пошук відрізка, на якому функція F змінює знак. Пошук у цьому алгоритмі виконується у напрямку антиградієнта для внутрішніх точок та градієнта для зовнішніх. Потім за допомогою двійкового пошуку (алгоритм `binary`) знаходиться точка на межі Ω .

Алгоритм двійкового пошуку `binary` знаходить корінь функції F на відріжку $[v_1; v_2]$. Його можна записати у такому вигляді:

```
algorithm binary
input:
     $v_1$  – координати точки 1
     $v_2$  – координати точки 2
     $F$  – функція, що змінює знак між точками  $v_1$  і  $v_2$ 
     $\varepsilon$  – чисельний нуль
output:
     $b$  – точка з відріжку  $[v_1; v_2]$ , яка є коренем  $F$ 
begin
     $f_1 \leftarrow F(v_1)$ 
     $f_2 \leftarrow F(v_2)$ 
    do
         $b \leftarrow 0,5 (v_1 + v_2)$ 
         $f_b \leftarrow F(b)$ 
        if  $\text{sign}(f_1) \neq \text{sign}(f_b)$  then
             $v_2 \leftarrow b$ 
             $f_2 = f_b$ 
        else
             $v_1 \leftarrow b$ 
             $f_1 \leftarrow f_b$ 
        end if
    while  $|f_b| \geq \varepsilon$ 
return  $b$ 
```

У результаті застосування алгоритму `background_grid_surface` буде отримана дискретна модель межі об'єкта. Структура граней дискретної моделі визначається початковою сіткою та функцією F . Отримані дискретні моделі меж об'єктів, зазвичай, вимагають додаткового згладжування та оптимізації положень вузлів для отримання найкращого результату.

1.10.1.3 Приклади застосування

Розглянемо форму об'єкта, задану такою функцією:

$$w(x, y, z) = \text{cuboid}(x, y, z, 2, 2, 2) \wedge \text{sphere}(x, y, z, 1) \wedge \neg \text{circle}(x; y; 0,5) \wedge \neg \text{circle}(x; z; 0,5) \wedge \neg \text{circle}(y; z; 0,5), \quad (1.58)$$

де $\text{cuboid}(x, y, z, w, h, d)$ – функція (1.28);

$\text{sphere}(x, y, z, r)$ – функція (1.24);

$\text{circle}(x, y, r)$ – функція (1.11).

У результаті використання рівномірної фонові сітки, що складається з 9261 вузла та 48000 тетраедрів, яка побудована на кубі $[-1; 1] \times [-1; 1] \times [-1; 1]$ (рис. 1.65, а), після видалення зовнішніх вузлів залишиться 2158 межових вузлів і 4332 елемента (рис. 1.65, б). Потім після знаходження відповідних проєкцій на межу буде отримана дискретна модель межі у вигляді рис. 1.65, в.

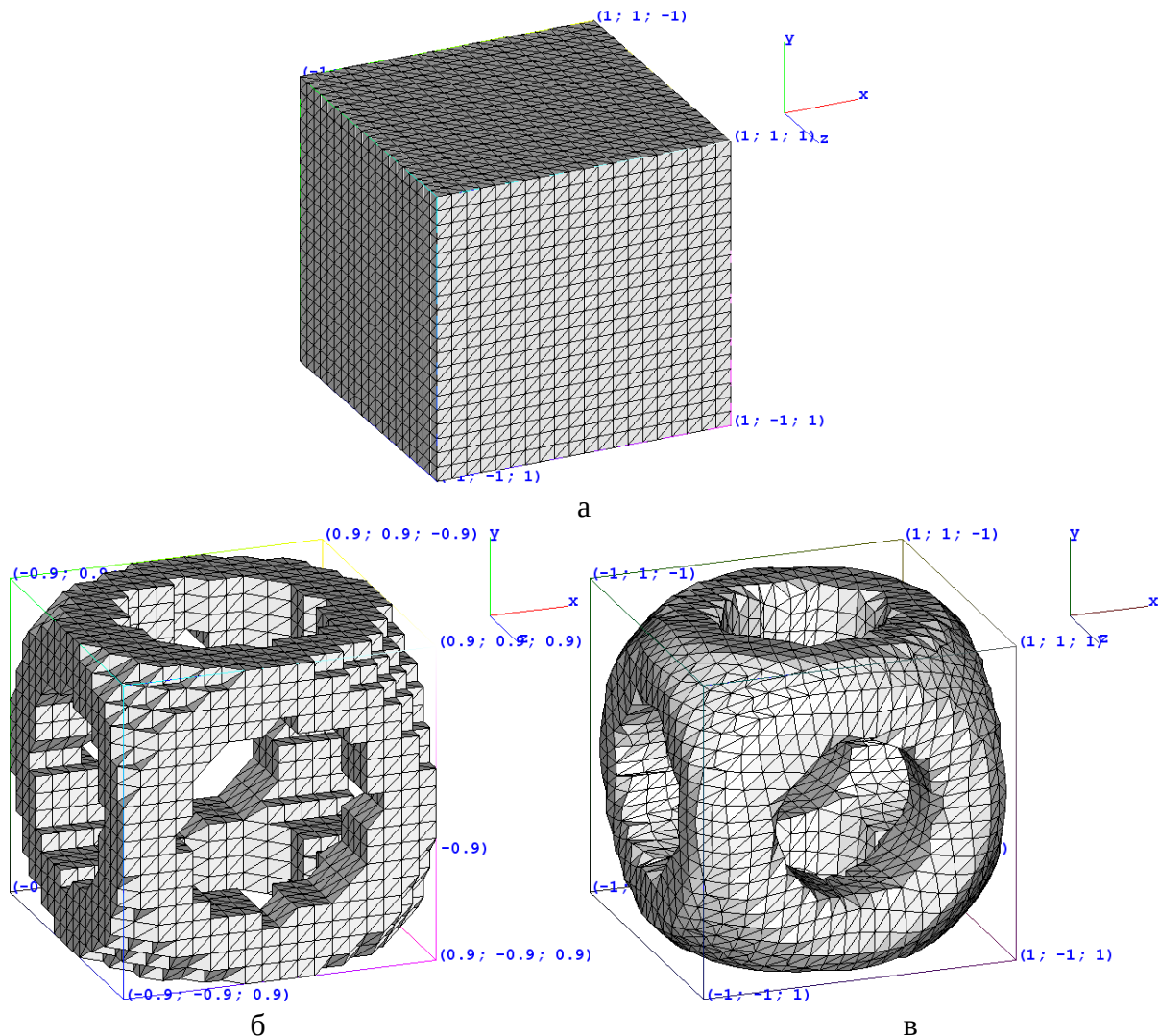


Рисунок 1.65 – Приклад побудови дискретної моделі межі з використанням фонові сітки тетраедрів для об'єкта, заданого формулою (1.58): а – фонові сітка тетраедрів; б – фонові сітка тетраедрів після видалення зовнішніх вузлів; в – дискретна модель на базі трикутників на межі об'єкта

Для отримання дискретної моделі межі об'єкта, заданого формулою (1.58), також можна використовувати фонові сітку, яка, наприклад, складається з 9261 вузла і 8000 кубічних елементів (рис. 1.66, а). Після видалення зовнішніх вузлів з дискретної моделі залишиться 3064 граничних вузла і 3072 елементів

(рис. 1.66, б), проєкція яких на межу дозволить отримати результат (рис. 1.66, в).

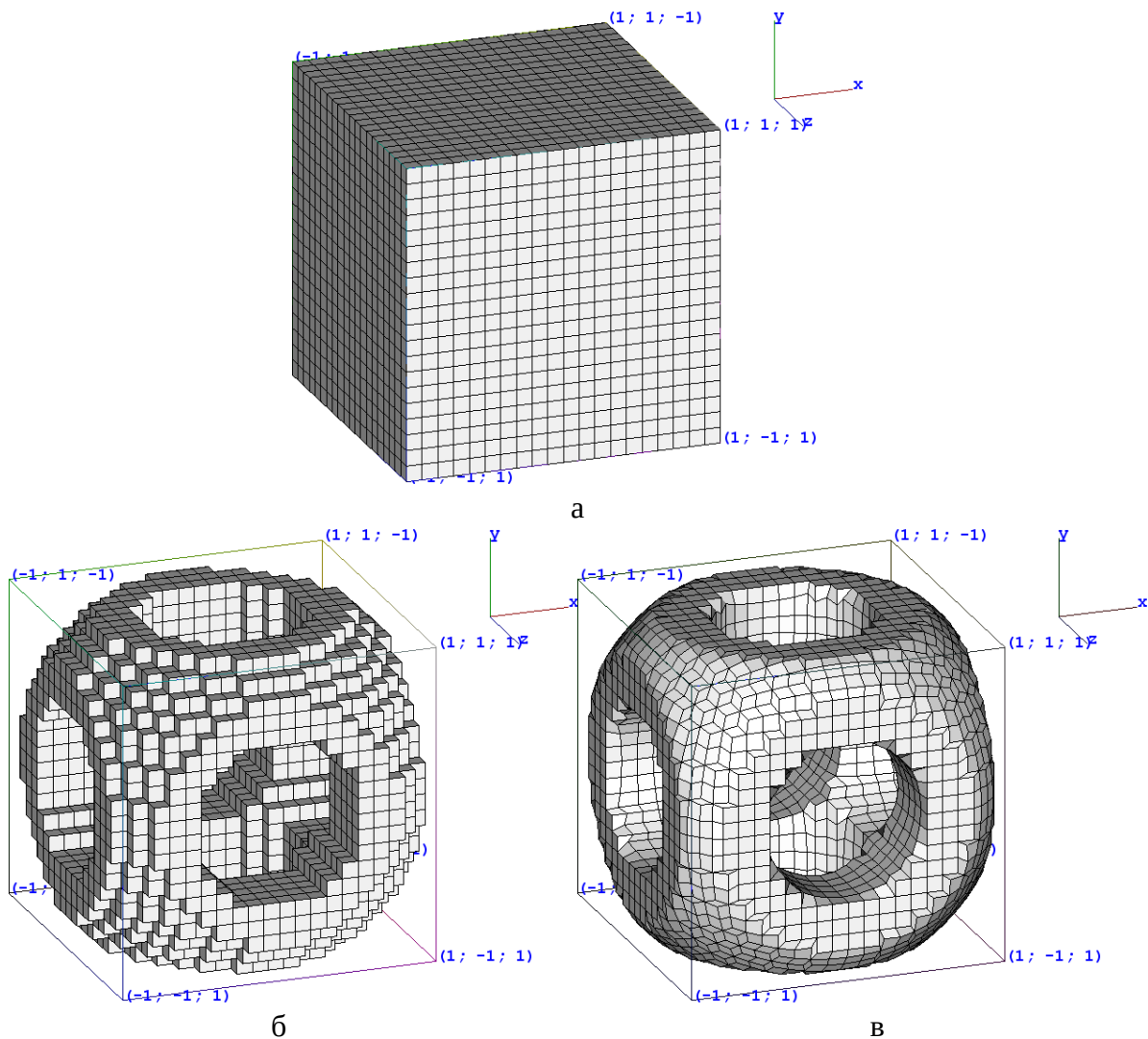


Рисунок 1.66 – Приклад побудови дискретної моделі межі з використанням фонові сітки кубів для геометричного об'єкта, заданого формулою (1.58): а – фонові сітка кубів; б – фонові сітка кубів після видалення зовнішніх вузлів; в – дискретна модель на базі чотирикутників на межі об'єкта

Як фонові також можна використовувати неструктуровані дискретні моделі форм об'єктів. Так, наприклад, для моделі пробки, поданої формулою (1.33), фонові може виступати дискретна модель кругового циліндра з радіусом основи, що дорівнює 25, і висотою 50 (рис. 1.67, а для тетраєдрів і рис. 1.67, б для шестигранників). В результаті застосування алгоритму `background_grid_surface` для фонових моделей циліндричної області будуть отримані дискретні моделі межі пробки (рис. 1.67).

Отже, густина вузлів та елементів у підсумкових дискретних моделях визначається наявністю відповідних областей у фонових сітках. Тобто, при використанні рівномірних фонових сіток будуть отримані близькі до

Математичне забезпечення інженерного аналізу об'єктів аерокосмічної техніки на базі хмарних технологій 103

рівномірних моделі меж, а для адаптивних фонових дискретних моделей буде більш помітною зміна щільності вузлів по поверхні.

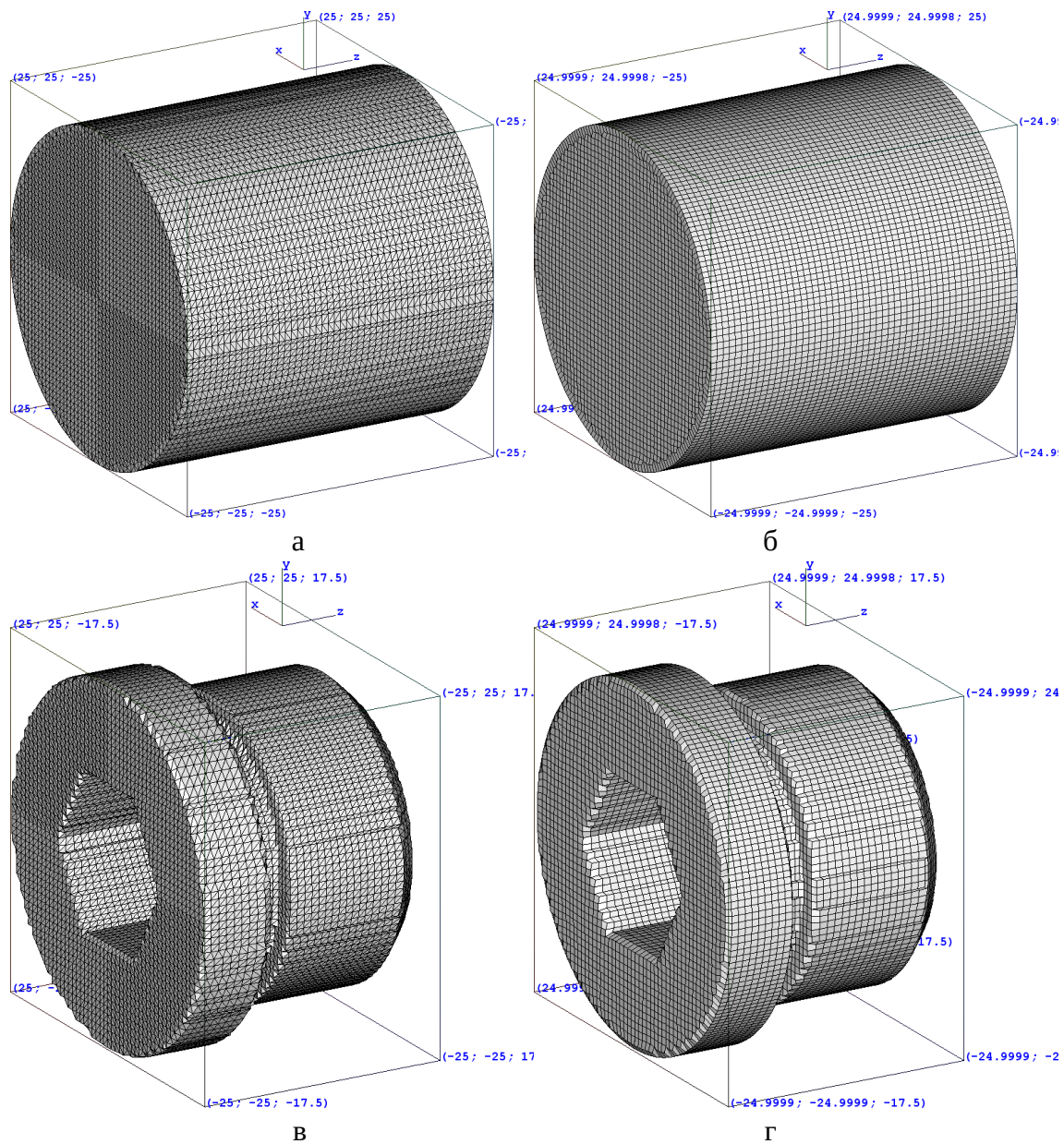


Рисунок 1.67 – Дискретні моделі межі пробки: а – фонові сітка тетраедрів (103275 вузлів і 586200 елементів); б – фонові сітка шестигранників (108783 вузла і 101800 елементів); в – дискретна модель на базі трикутників (11705 вузлів і 23406 елементів); г – дискретна модель на базі чотирикутників (13070 вузлів і 13068 елементів)

Наприклад, для побудови дискретної моделі межі важеля передньої підвіски, заданого формулою (1.32), як фонову можна використовувати модель профілю, адаптивну до його меж, зі збільшенням кількості елементів біля отворів. У результаті застосування фонові дискретної моделі, яка складається з 39675 вузлів і 197424 тетраедрів, можна отримати 7983 вузла і 15974 трикутника на межі (рис. 1.68).

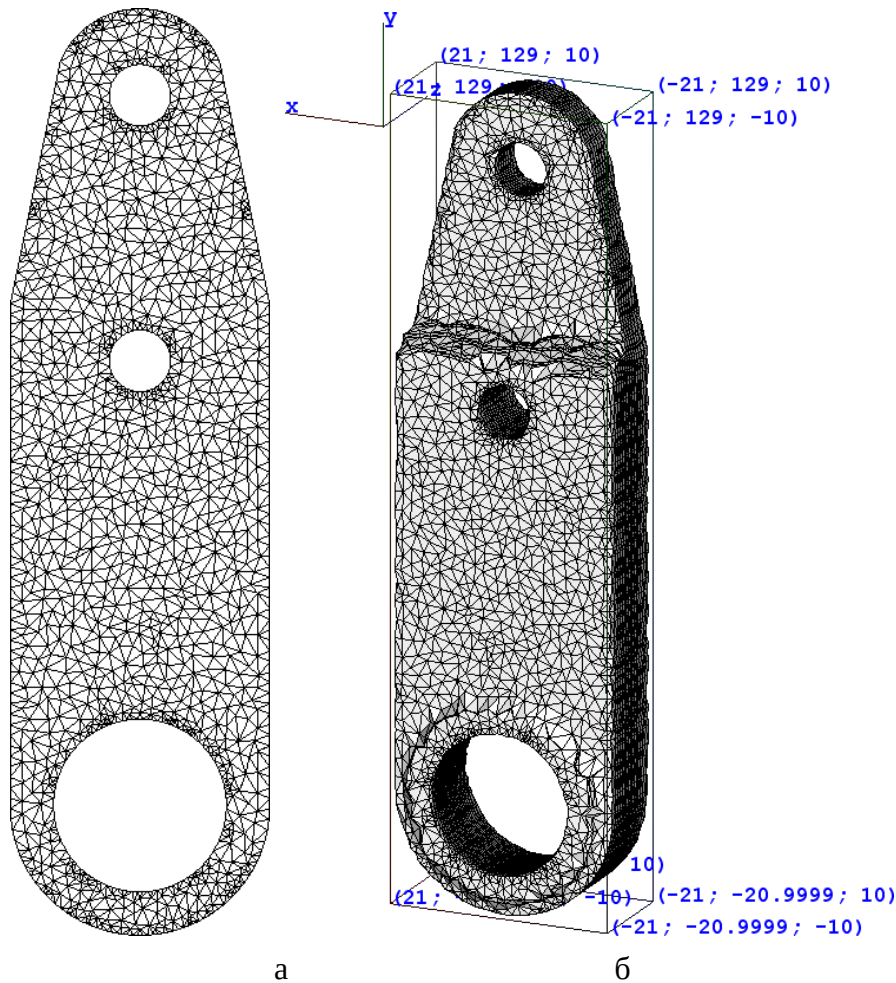


Рисунок 1.68 – Приклад використання адаптивної фонові дискретної моделі:

а – фонові сітка тетраєдрів;

б – трикутники на межі моделі подовжувача важеля передньої підвіски

1.10.1.4 Двовимірний випадок

Алгоритм `background_grid_surface`, у окремому випадку, може також застосовуватися для побудови контурів двовимірних областей. У цьому випадку на місці граней для пошуку проекцій виступатимуть сторони двовимірних елементів, а результатом завжди буде дискретна модель, яка подає контур відрізками прямої. Водночас описане вище зауваження про адаптивність залишається справедливим.

Наприклад, при використанні фонові сітки, яка складається з 676 вузлів і 1250 трикутників (рис. 1.69, а), для моделі пластини, заданої формулою (1.18), буде отримана модель межі, яка складається з 123 вузлів і 123 відрізків (рис. 1.69, б). Водночас при використанні фонові сітки, яка складається з 676 вузлів і 625 чотирикутників (рис. 1.69, в), буде отримана модель, яка складається з 132 вузлів і 132 відрізків (рис. 1.69, г).

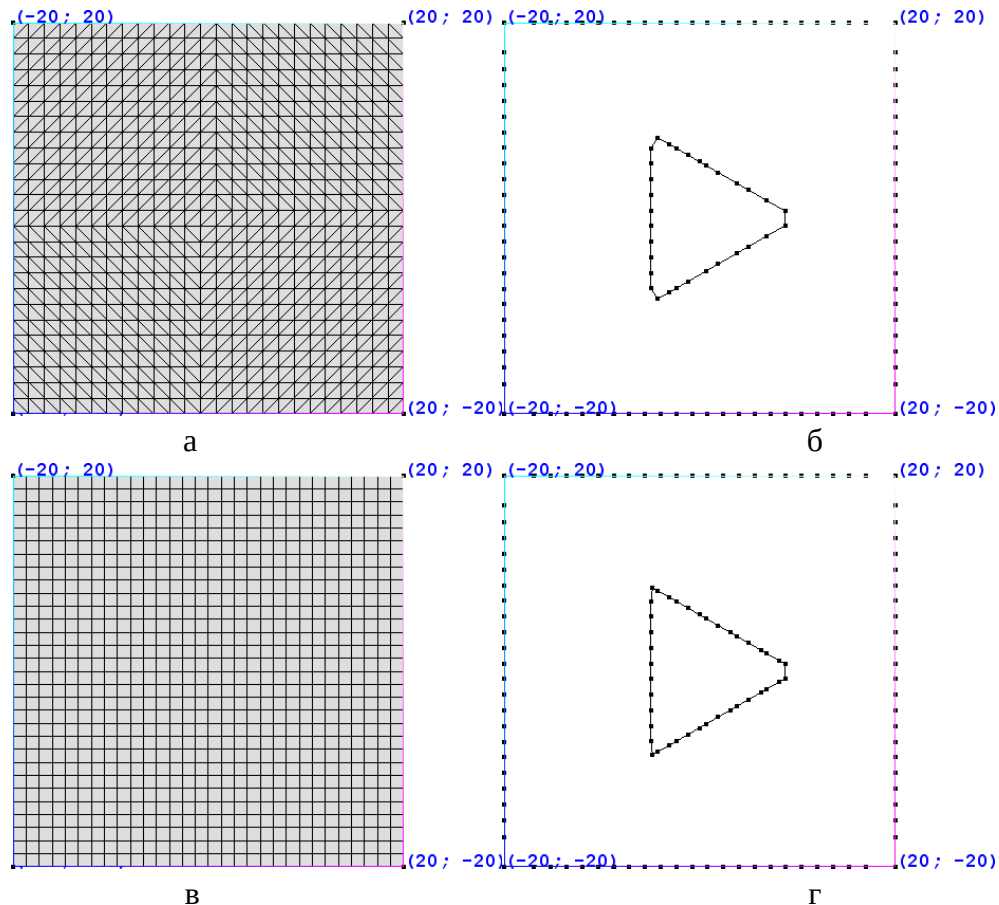


Рисунок 1.69 – Приклади побудови моделей меж двовимірної пластини: а – фонові сітка трикутників; б – модель межі для фонові сітки трикутників; в – фонові сітка чотирикутників; г – модель межі для фонові сітки чотирикутників

За аналогією з тривимірним випадком у двовимірному випадку необхідно використовувати нерівномірні моделі на базі плоских елементів для забезпечення адаптивності підсумкових моделей меж.

1.10.1.5 Аналіз результатів методу

На відміну від методів побудови дискретних моделей меж об'єктів, які ґрунтуються на скануванні рівномірними комірками [81, 297, 299, 300], розроблений метод теоретично дозволяє генерувати моделі на базі комірок довільної форми.

Розроблений метод і пов'язані з ним алгоритми не залежать від форми елементів. Вони можуть застосовуватися для генерації дискретних моделей на базі елементів різної форми. Єдина послідовність дій дозволяє проектувати програмне забезпечення на базі поведінкового шаблону проектування «Стратегія» [303].

Також запропонований метод можна використовувати для побудови картин ліній або поверхонь рівня для неявно заданих геометричних областей. Для цього у розроблених алгоритмах замість значення функції області необхідно аналізувати різницю її значення і константи, що відповідає необхідній лінії або поверхні рівня). Наприклад, для моделі пластини, заданої формулою (1.18), отримані лінії рівня, наведені на рис. 1.70.

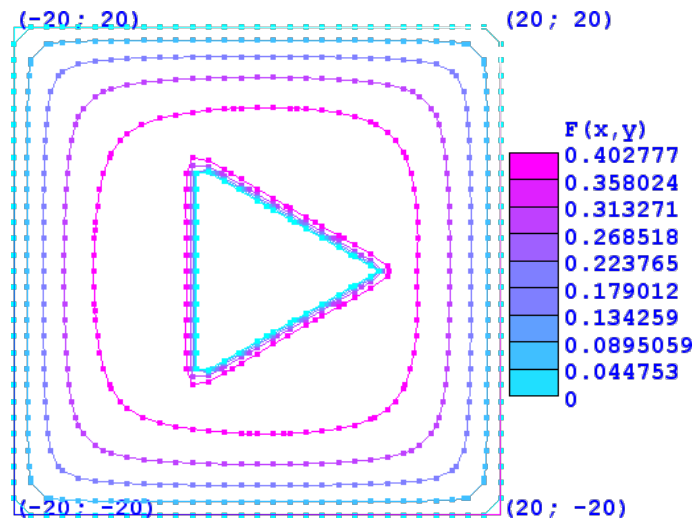


Рисунок 1.70 – Приклад ліній рівня для моделі двовимірної пластини

Аналіз отриманих дискретних моделей дозволяє зробити висновок про те, що застосування запропонованого методу може спричинити втрати точності апроксимації межі в околі її геометричних особливостей. Отже, до отриманих дискретних моделей необхідно застосовувати додаткові процедури їх оптимізації.

1.10.2 Метод згладжування координат вузлів дискретних моделей меж об'єктів, форма яких задана функціонально

Використання методів, що існують, [81, 232, 297, 298] та описаного вище підходу до генерації дискретних моделей меж неявних функцій може спричинити неточності апроксимації межі в околах її геометричних особливостей (наприклад, зламів поверхні). Збільшення числа вузлів та елементів у таких областях дозволяє досягти прийнятних результатів з точки зору візуалізації, але зумовлює значне зростання кількості невідомих, якщо моделі будуть використані у чисельному аналізі. У результаті актуальною є розробка методів підвищення точності апроксимації в околах зламів для дискретних моделей меж об'єктів, які не вимагають збільшення числа вузлів.

Аналіз результатів досліджень, присвячених оптимізації межових вузлів та елементів (пункт 1.5.2), дозволяє зробити висновок, що розроблені методи вирішують задачу точності апроксимації меж в околицях зламів для дискретних моделей, заснованих переважно на трикутниках. Підходи та методи, запропоновані, наприклад, у працях [160, 161, 227, 228, 232, 265, 266, 272, 274, 275, 298], потребують додавання нових вузлів у дискретну модель, що спричиняє практичну неможливість їх застосування для чотирикутників. Використання ієрархічних структур за схемою, яка запропонована у роботах [224, 226, 267, 268, 270, 271, 282], дозволяє отримати прийнятну якість. Проте, водночас буде істотно зростати кількість вузлів та елементів, що буде спричиняти збільшення розмірності задачі (наприклад, у додатках до скінченно-елементного аналізу). Сегментація та кластеризація [276, 277] дозволяють виявити зони, у яких є геометричні особливості, але також вимагають вставки додаткових вузлів. Отже, у опублікованих на даний момент роботах не виявлено ефективні методи оптимізації дискретних моделей, які можна застосовувати одночасно і до трикутників, і до чотирикутників.

1.10.2.1 Функціонал відстані-довжини

Одним з найбільш вживаних та водночас простих методів покращення дискретних моделей є локальне згладжування Лапласа [244]. Воно пересуває кожен вузол у геометричний центр багатокутника, утвореного суміжними вузлами (рис. 1.71). Традиційно для досягнення кращого результату це згладжування застосовують кілька разів.

У згладжуванні Лапласа кожне ребро, яке сполучене з центральним вузлом, можна розглядати як пружину з початковою нульовою довжиною [265]. Сума сил, що діють у i -му вузлі сітки $\mathbf{M}$, визначається формулою:

$$F(\mathbf{M}, i, \kappa) = \kappa \sum_{k \in A_i} (\mathbf{M}[k] - \mathbf{M}[i]),$$

де κ – жорсткість пружини;

$A_i = \mathbf{M}.\text{adjacent}(i)$ – множина кодів суміжних з i -м вузлів.

Точка рівноваги системи пружин відповідає мінімуму потенційної енергії, яка визначається такою формулою:

$$E(\mathbf{M}, i, \kappa) = \frac{\kappa}{2} \sum_{k \in A_i} \|\mathbf{M}[k] - \mathbf{M}[i]\|^2. \quad (1.59)$$

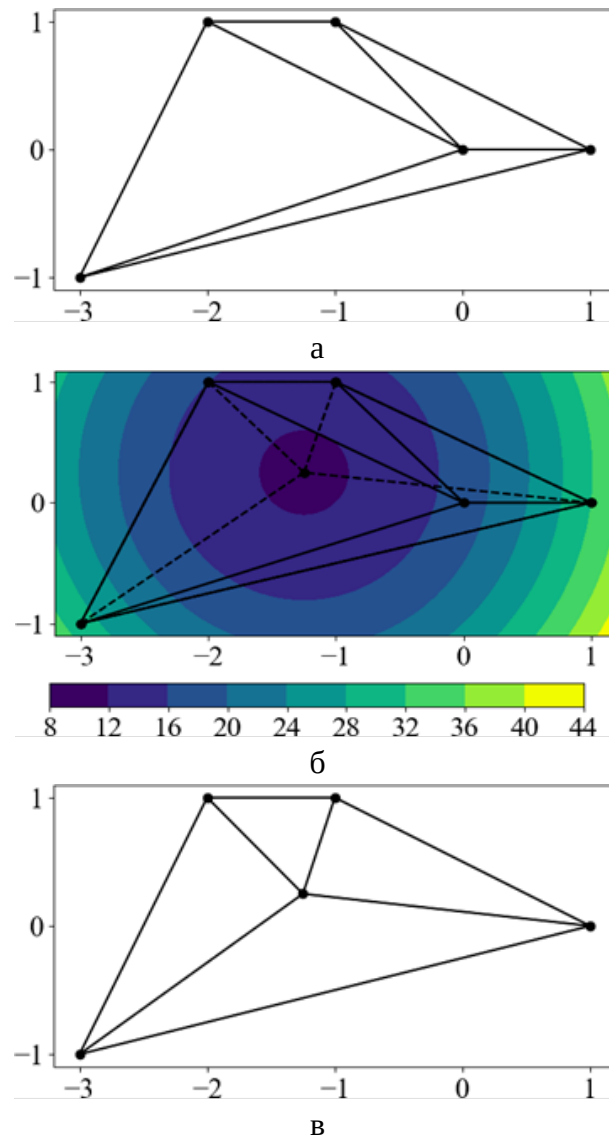


Рисунок 1.71 – Згладжування Лапласа: а – вихідна сітка; б – розподіл значень функції (1.59); в – результат згладжування

Формулу (1.59) у тривимірному випадку можна записати у такому вигляді:

$$E(\mathbf{M}, i, \kappa) = \frac{\kappa}{2} \sum_{k \in A_i} \left((x_k - x_i)^2 + (y_k - y_i)^2 + (z_k - z_i)^2 \right), \quad (1.60)$$

де x_i, y_i і z_i – відповідно, абсциса, ордината та апліката i -го вузла;

x_k, y_k і z_k – відповідно, абсциса, ордината та апліката k -го вузла.

Диференціюючи функцію (1.60) за x_i, y_i, z_i та порівнюючи до нуля отримаємо

$$x_i = \frac{1}{|A_i|} \sum_{k \in A_i} x_k,$$

$$y_i = \frac{1}{|A_i|} \sum_{k \in A_i} y_k,$$

$$z_i = \frac{1}{|A_i|} \sum_{k \in A_i} z_k.$$

де $|A_i|$ – кількість суміжних з i -м вузлів.

У результаті виходить проста та дуже ефективна (з обчислювальної точки зору) процедура. Її застосування для дискретних моделей меж об'єктів пов'язане з пошуком проекції нової позиції точки на межу. Багаторазове ітераційне застосування такого згладжування прагне забезпечити рівну довжину сторонам елементів. Проте, відсутність інформації про точність апроксимації, зазвичай, зумовлює збільшення похибки в околах особливостей поверхні.

Наприклад, для дискретної моделі кулі одиничного радіуса з осьовим круговим отвором (радіус отвору дорівнює половині радіуса кулі) для вихідної дискретної моделі на базі трикутників (рис. 1.72, а) згладжена модель матиме вигляд, наведений на рис. 1.72, б. Аналогічно погіршується точність апроксимації межі для чотирикутників (рис. 1.72, в, г).

Розглянемо енергетичну функцію вигляду

$$E(\mathbf{M}, i, \kappa, F) = \kappa \sum_{k \in A_i} \|\mathbf{M}[k] - \mathbf{M}[i]\|^2 + (1 - \kappa) \sum_{q \in I_i} \|C_q - \text{projection}(C_q, F)\|^2, \quad (1.61)$$

де κ – параметр методу;

F – неявна функція, що описує форму об'єкта;

A_i – множина кодів суміжних з i -м вузлів;

$I_i = \mathbf{M}.\text{incident}(i)$ – множина кодів інцидентних до i -го вузла елементів;

$C_q = \frac{1}{|\mathbf{M}(q)|} \sum_{j \in \mathbf{M}(q)} \mathbf{M}[j]$ – центр мас елемента з кодом q ;

$\text{projection}(C_q, F)$ – проекція точки C_q на поверхню, яка задана функцією F .

Перший доданок у формулі (1.61) гарантує існування мінімуму [265]. Вибір другого доданка виконаний з наступного припущення. Кожен плоский елемент дискретної моделі межі об'єкта можна вважати апроксимацією відповідної частини поверхні (рис. 1.73), а відстань від центру елемента до поверхні – показником точності такої апроксимації.

При обчисленні значення формули (1.61) пошук проекції точки на поверхню можна здійснювати у напрямку градієнта (для зовнішніх точок) або антиградієнта (для внутрішніх) за допомогою алгоритму `find_border`. Використання довжин сторін елементів та відстаней від центрів елементів до поверхні обумовлює назву «функціонал відстані-довжини».

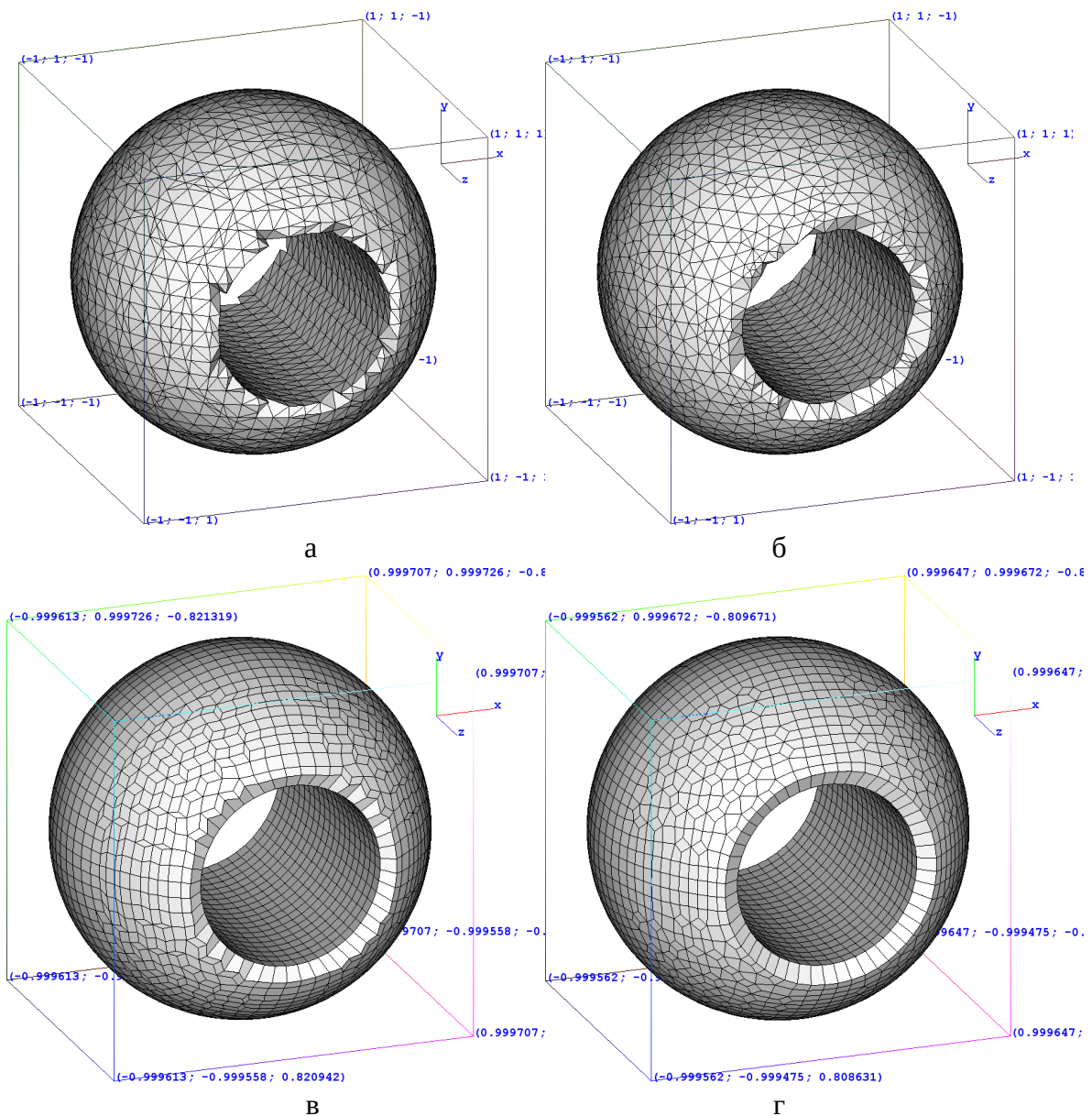


Рисунок 1.72 – Приклади застосування згладжування Лапласа: а – вихідна дискретна модель трикутників; б – результат згладжування дискретної моделі трикутників; в – вихідна дискретна модель чотирикутників; г – результат згладжування дискретної моделі чотирикутників

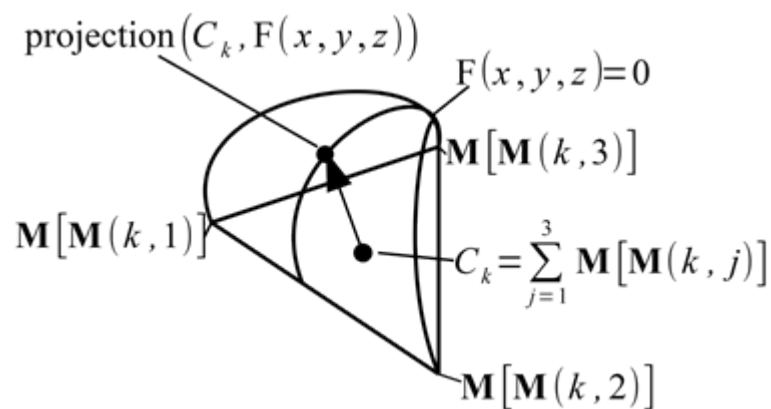


Рисунок 1.73 – Апроксимація частини поверхні трикутником

1.10.2.2 Локальний алгоритм пошуку мінімуму функціоналу відстані-довжини

Знаходження аналітичного виразу для точки мінімуму функціоналу (1.61), подібно до того, як це зроблено для формули (1.60) не видається можливим через неявність i , зазвичай, нелінійність функції $F(x, y, z)$. Для чисельного пошуку мінімуму можна скористатися модифікацією методу Гауса, яка формалізована у вигляді такого алгоритму:

algorithm local_refinement

input:

F – функція, яка подає форму об'єкта

$\mathbf{M}$ – дискретна модель межі об'єкта

k_n – код вузла для оптимізації

κ – параметр жорсткості системи

ε – чисельний нуль

output:

$\mathbf{M}$ – дискретна модель межі об'єкта

begin

$\mathbf{A}_k \leftarrow \mathbf{M}.\text{adjacent}(k_n)$

$P_0 \leftarrow \mathbf{M}[k_n]$

$f_0 \leftarrow E(\mathbf{M}, k_n, \kappa, F)$

for each $j \in \mathbf{A}_k$ **do**

$l \leftarrow 0,1$

do

$P_1 \leftarrow (1 - l) \mathbf{M}[k_n] + l \mathbf{M}[j]$

$h \leftarrow \|P_0 - P_1\|$

$\mathbf{M}[k_n] \leftarrow \text{find\_border}(P_1, F, h, \varepsilon)$

$f_1 \leftarrow E(\mathbf{M}, k_n, \kappa, F)$

if $f_0 \leq f_1$ **then**

$\mathbf{M}[k_n] \leftarrow P_0$

$l \leftarrow \frac{l}{2}$

else

$f_0 \leftarrow f_1$

end if

while $l \geq \varepsilon$

end for

end

Пошук здійснюється у l -координатах багатокутника, який утворений сусідніми з i -м вузлами. Спуск здійснюється по кожній з l -координат, поки

цільова функція спадає. Використовується адаптивний крок: у кожному напрямку, починаючи з кроку, значення якого емпірично прийнято рівним 0,1, якщо цільова функція перестає спадати, тоді крок зменшується вдвічі.

1.10.2.3 Глобальний алгоритм пошуку мінімуму функціоналу відстані-довжини

Застосування локального алгоритму послідовно до кожного вузла дискретної моделі межі об'єкта можна розглядати як ітераційне наближення до мінімуму функціоналу (1.61) у глобальній постановці. Відповідно, необхідно послідовно застосовувати локальний алгоритм для кожного вузла дискретної моделі межі об'єкта. В результаті отримаємо такий алгоритм оптимізації положень вузлів дискретної моделі межі об'єкта, форма якого задана функціонально:

```
algorithm boundary_refinement
input :
    F – функція, яка подає форму об'єкта
    M – дискретна модель межі об'єкта
    κ – параметр жорсткості системи
    ε – чисельний нуль
output :
    M – дискретна модель межі об'єкта
begin
    for each  $n \in M.N$  do
        local_refinement(F, M, M.code(n), κ, ε)
    end for
end
```

За аналогією зі згладжуванням Лапласа алгоритм boundary_refinement необхідно застосовувати кілька разів. Водночас на кожній ітерації додатково можуть використовуватися методи оптимізації дискретних моделей, які ґрунтуються на локальних перебудовах елементів. Наприклад, при використанні на кожній ітерації операції «flip» [263, 264, 275] для трикутників можна гарантувати, що дискретна модель буде відповідати критерію Делоне.

1.10.2.4 Приклади застосування методу

Розглянемо результати застосування алгоритму пошуку мінімуму функціоналу відстані-довжини, для оптимізації дискретної моделі, яка зображена на рис. 1.60, а. На рис. 1.74 наведено результати застосування алгоритму, якщо $\kappa = 10^{-3}$ і $\varepsilon = 10^{-6}$. Колір елемента показує значення найменшого кута у градусах (вважається, що чим більшим є мінімальний кут, тим якісніша модель). Можна помітити, що, починаючи з чотирьох ітерацій, злам поверхні біля отвору апроксимується вузлами та ребрами елементів, але при цьому залишаються гострі (менше 10 градусів) кути.

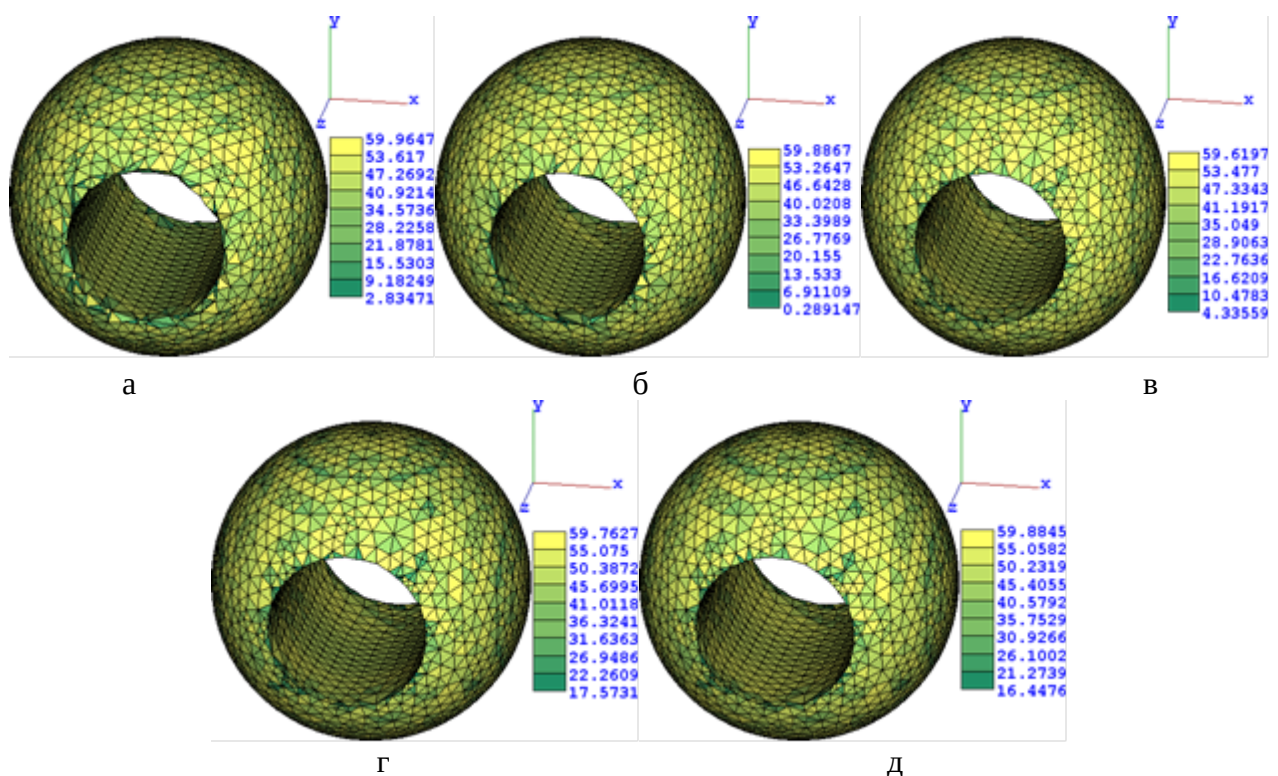


Рисунок 1.74 – Застосування локальної мінімізації функціоналу відстані-довжини для оптимізації положень вузлів у моделі кулі з круговим отвором (трикутники):
а – 1 ітерація; б – 2 ітерації; в – 4 ітерації; г – 8 ітерацій; д – 16 ітерацій

Запропонований алгоритм може застосовуватися також для покращення дискретних моделей меж об'єктів на базі чотирикутних елементів. Наприклад, для дискретної моделі, яка зображена на рис. 1.60, в, результати застосування запропонованого алгоритму, якщо $\kappa = 10^{-3}$ і $\varepsilon = 10^{-6}$, матимуть вигляд, показаний на рис. 1.75. Колір елемента показує значення найбільшого кута у градусах (у випадку чотирикутників, чим більшим є максимальний кут, тим ближче він до виродженого елемента). Як й у випадку трикутників, починаючи з чотирьох ітерацій, злам поверхні біля отвору апроксимується вузлами та ребрами елементів, але залишаються елементи, які за формою близькі до трикутниках.

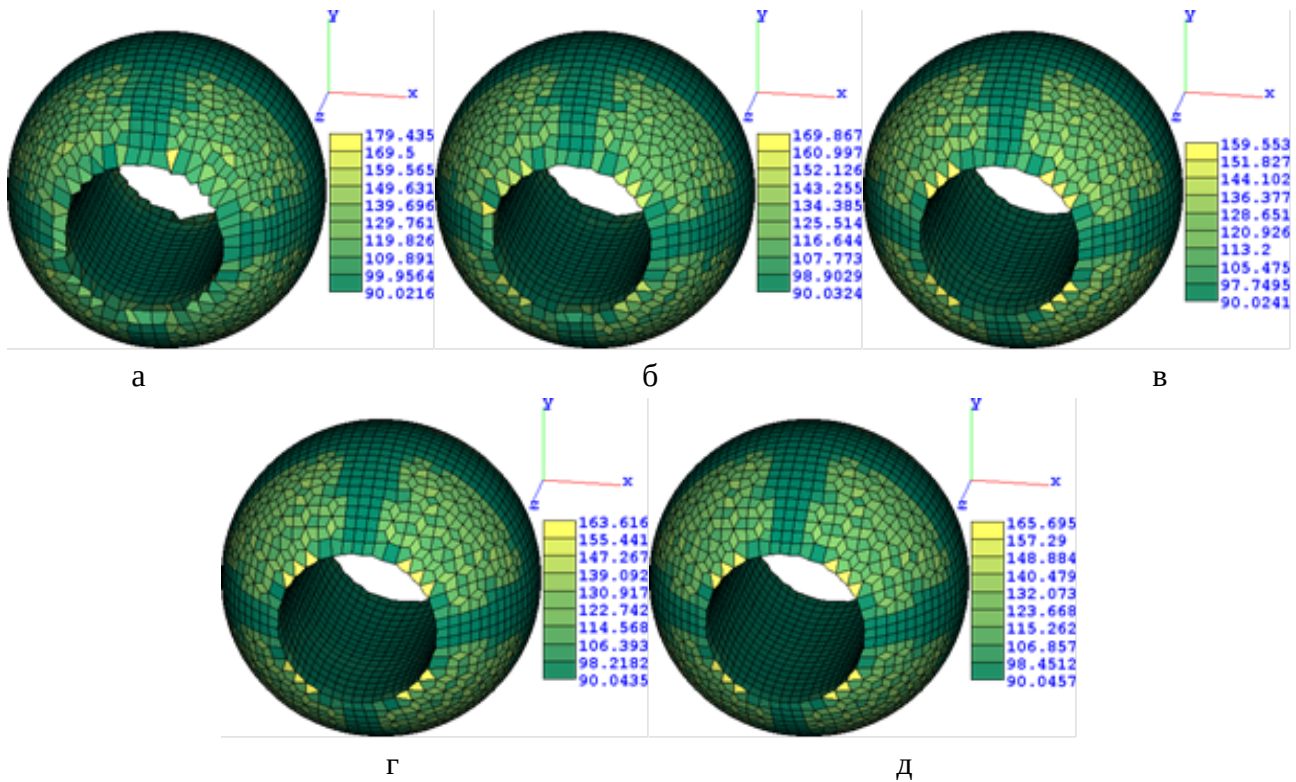


Рисунок 1.75 – Застосування локальної мінімізації функціоналу відстані-довжини для оптимізації положень вузлів у моделі кулі з круговим отвором (чотирикутники): а – 1 ітерація; б – 2 ітерації; в – 4 ітерації; г – 8 ітерацій; д – 16 ітерацій

При дослідженні впливу параметра k другий доданок у формулі (1.61) можна розглядати як силу тяжіння центрів елементів до поверхні. Тоді жорсткість k визначає можливість «опору» системи пружин між вузлами руху вузлів по поверхні. Відповідно, чим більше k , тим буде ближче результат мінімізації до згладжування Лапласа, а зі зменшенням значення k більш точно будуть апроксимувати злами, але можливе отримання близько розташованих один до одного вузлів. На рис. 1.76 показані результати варіювання k для 8 ітерацій покращення сітки.

Модель кубика зі стороною рівною 2, у якому два кругових отвори (твірні, паралельні Ox і Oz) радіуса 0,5 і циліндричні ручки (твірні, паралельні Oz) того само радіуса й довжиною 1 (рис. 1.77), можна задати такою формулою:

$$w(x, y, z) = [\text{cuboid}(x, y, z, 2, 2, 2) \vee (\text{circle}(x; z; 0,5) \wedge \text{stripe}(y, 4))] \wedge (\neg \text{circle}(x; y; 0,5) \wedge \neg \text{circle}(y; z; 0,5)), \quad (1.62)$$

де cuboid – функція (1.28);

circle – функція (1.11);

stripe – функція (1.15).

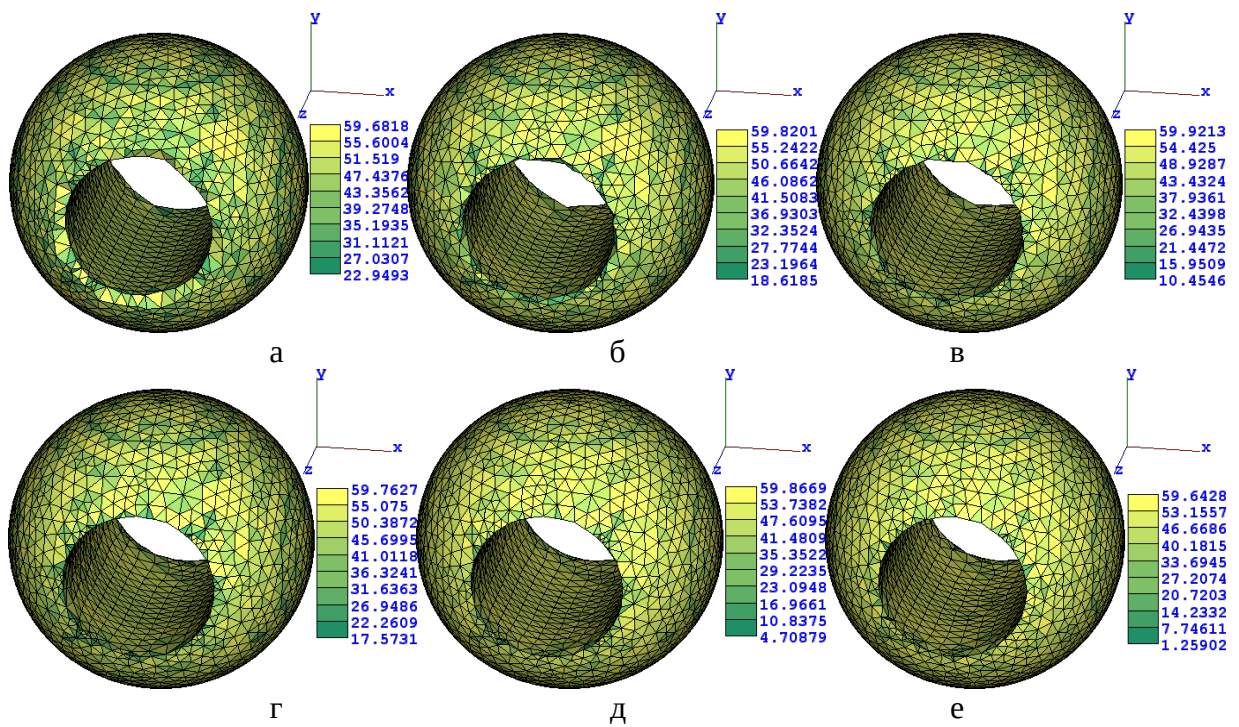


Рисунок 1.76 – Вплив параметра κ :

а – $\kappa = 0,1$; б – $\kappa = 0,01$; в – $\kappa = 0,005$; г – $\kappa = 0,001$; д – $\kappa = 0,0005$; е – $\kappa = 0,0001$

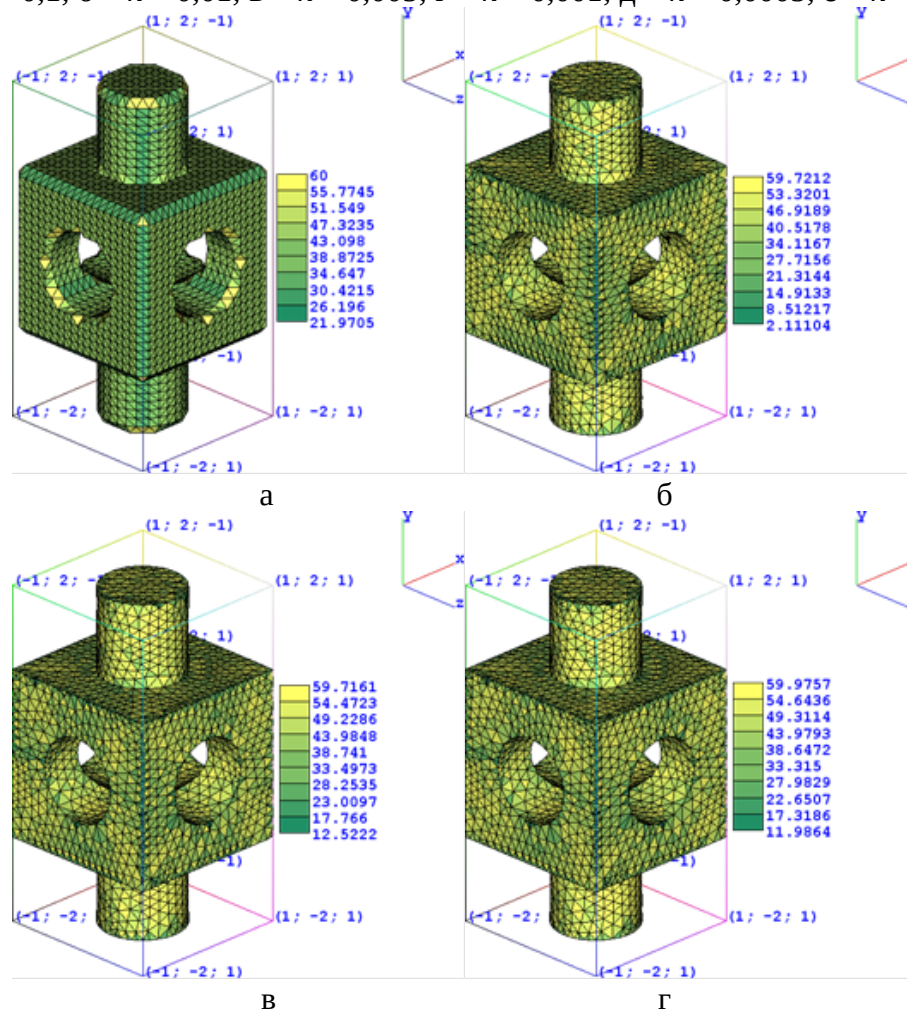


Рисунок 1.77 – Застосування локальної мінімізації функціоналу відстані-довжини для оптимізації положень вузлів у моделі «кубик з ручками» (трикутники):
а – вихідна дискретна модель; б – 4 ітерації оптимізації; в – 8 ітерацій оптимізації;
г – 16 ітерацій оптимізації

На рис. 1.77, а показана вихідна дискретна модель на базі трикутних елементів, а також результати застосування алгоритму локальної мінімізації функціоналу відстані-довжини, якщо $\kappa = 10^{-3}$ і $\varepsilon = 10^{-6}$ (рис. 1.77, б–г). Аналогічно, на рис. 1.78, а–г показані результати для покращення сітки чотирикутних елементів.

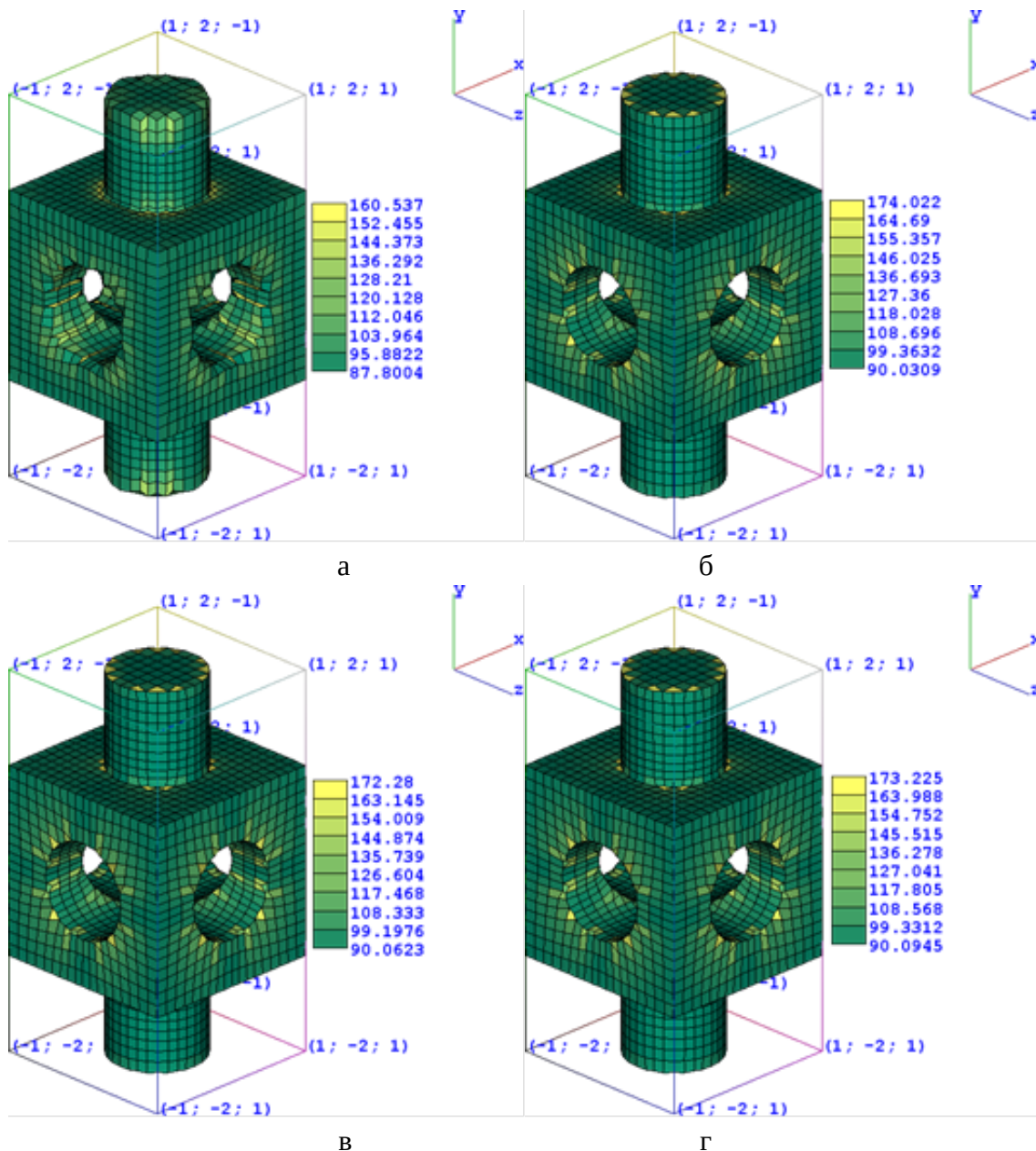


Рисунок 1.78 – Застосування локальної мінімізації функціоналу відстані-довжини для оптимізації положень вузлів у моделі «кубик з ручками» (чотирикутники): а – вихідна дискретна модель; б – 4 ітерації оптимізації; в – 8 ітерацій оптимізації; г – 16 ітерацій оптимізації

Запропонований алгоритм можна застосувати й до покращення моделей гладких поверхонь. Наприклад, для поверхні «genus3» [273], яку можна подати такою формулою:

$$\text{genus3}(x, y, z) = -r_z^4 z^2 + \left(1 - \frac{x^2}{r_x^2} - \frac{y^2}{r_y^2}\right) \times \\ \times ((x - x_1)^2 + y^2 - r_1^2)((x + x_1)^2 + y^2 - r_1^2)(x^2 + y^2 - r_1^2),$$

при значеннях параметрів: $r_x = 6$, $r_y = 3,5$, $r_z = 4$, $r_1 = 1,2$ і $x_1 = 3,9$, результати покращення дискретних моделей ($\kappa = 10^{-3}$ і $\varepsilon = 10^{-6}$) набудуть вигляд рис. 1.79.

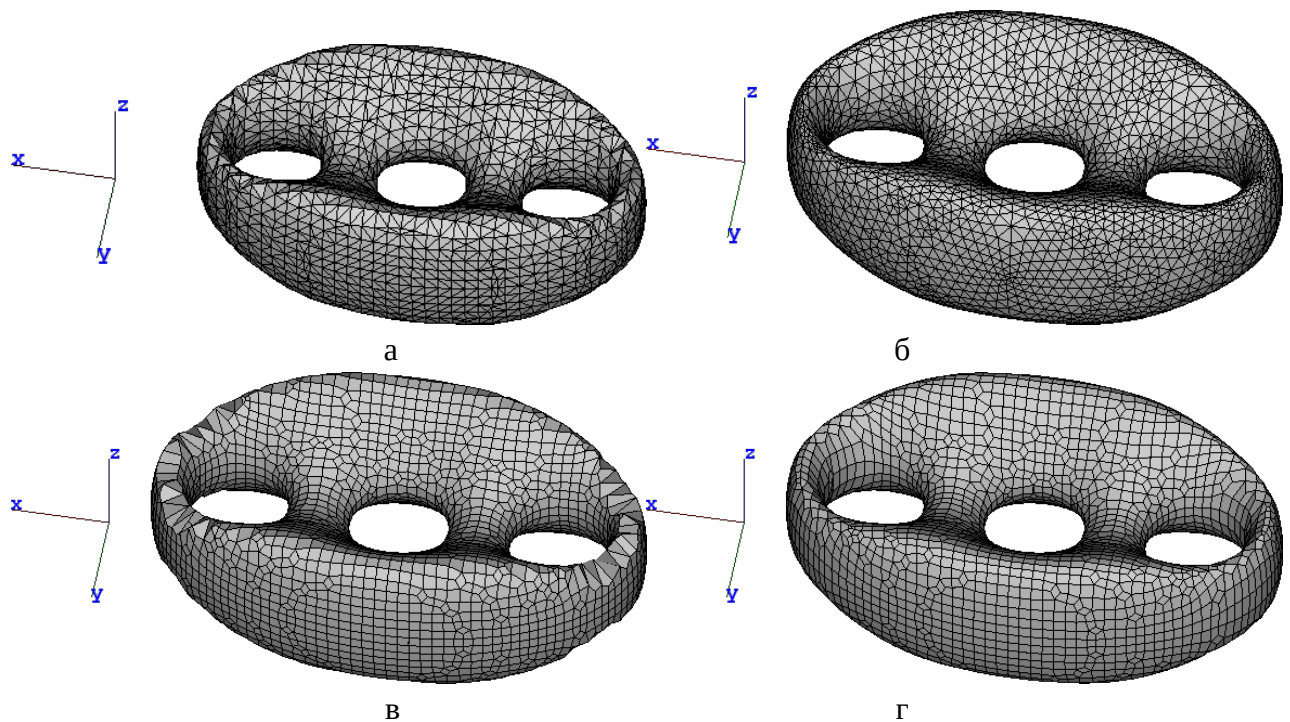


Рисунок 1.79 – Застосування локальної мінімізації функціоналу відстані-довжини для оптимізації положень вузлів у моделі «genus3»: а – вихідна дискретна модель на базі трикутників; б – 8 ітерацій оптимізації для моделі на базі трикутників; в – вихідна дискретна модель на базі чотирикутників; г – 8 ітерацій оптимізації для моделі на базі чотирикутників

Отже, застосування запропонованого алгоритму до дискретних моделей гладких поверхонь буде забезпечувати згущення числа вузлів у областях з найбільшою кривиною.

1.10.2.5 Двовимірний випадок

Описані вище функціонал і алгоритм у окремому випадку можуть застосовуватися для покращення меж форм двовимірних об'єктів. У такому випадку елементом межі виступає відрізок прямої, знаходження відстані до межі від якого дозволить обчислювати функціонал вигляду (1.61). Алгоритм

мінімізації функціоналу не потребує змін, тому що пошук виконується у l -координатах суміжних вузлів.

Наприклад, для моделі пластини, заданої формулою (1.18), якщо як вихідні використовувати дискретні моделі ліній рівня (рис. 1.58), то у результаті застосування 8 ітерацій оптимізації для кожної моделі лінії рівня, результат набуде вигляд, наведений на рис. 1.80.

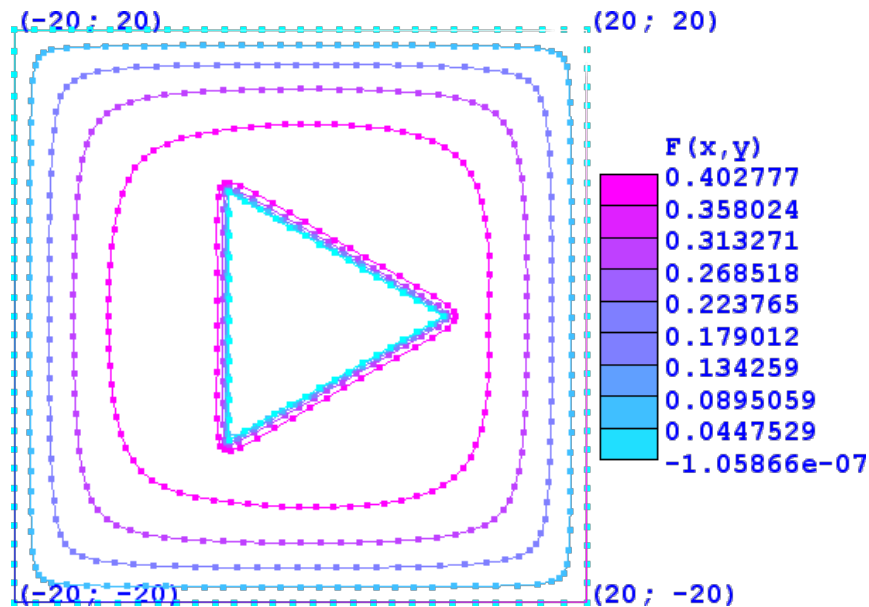


Рисунок 1.80 – Застосування локальної мінімізації функціоналу відстані-довжини для оптимізації положень вузлів у двовимірних дискретних моделях

1.10.2.6 Аналіз результатів методу

Отримані результати моделювання меж об'єктів (рис. 1.74–1.80) показують, що запропонований метод згладжування дискретних моделей меж об'єктів, який заснований на локальній мінімізації функціоналу відстані-довжини, дозволяє покращити точність дискретних моделей в околах особливостей (зламів поверхні). Водночас точність апроксимації особливостей буде залежати від параметра жорсткості ребер елементів вихідної дискретної моделі, що показано на рис. 1.74.

Перевага запропонованого метода полягає у тому, що (на відміну від результатів [224–225, 263–266]) він може застосовуватися і до трикутників, і до чотирикутників. Також метод не вимагає вставки додаткових вузлів і елементів.

Необхідно зазначити, що даний метод покращення дискретних моделей не враховує метрик якості (мінімальних або максимальних кутів, співвідношень довжин сторін тощо). У результаті як недолік можна зазначити можливість

одержання моделей з дуже гострими або тупими кутами. Також, як можна побачити з рис. 1.75 і 1.78 поблизу зламів поверхні можна отримати чотирикутники, за формою близькі до трикутників (один з кутів більше 175 градусів), усунути які без додаткового топологічного перебудування неможливо.

Запропонований метод можна використовувати спільно з методами локального згладжування для внутрішніх вузлів дискретних моделей форм об'єктів. У результаті можна сформулювати єдину процедуру для всіх вузлів моделі.

1.10.3 Метод локальних перебудов елементів дискретних моделей меж об'єктів

Описані вище методи генерації та згладжування дискретних моделей меж об'єктів використовують тільки операції переміщення вузлів без зміни топології елементів (зв'язків між вузлами). Проте, у багатьох випадках зміна зв'язків між вузлами дозволяє покращити характеристики дискретної моделі без втрати точності апроксимації геометричних особливостей межі об'єкта.

Наприклад, однією з найпростіших і водночас поширених операцій покращення дискретних моделей на базі трикутних елементів є «flip» [263, 275]. В основу цієї операції покладено ідею зміни спільного ребра (або діагоналі чотирикутника, утвореного вузлами) для пари сусідніх трикутників (рис. 1.81). «Flip» застосовують для приведення дискретних моделей у відповідність критерію Делоне [144], однією з важливих властивостей якого є максимізація мінімальних кутів у трикутниках моделі. У такому випадку діагональ змінюють, якщо описане коло одного трикутника включає у себе всі вершини сусіднього.

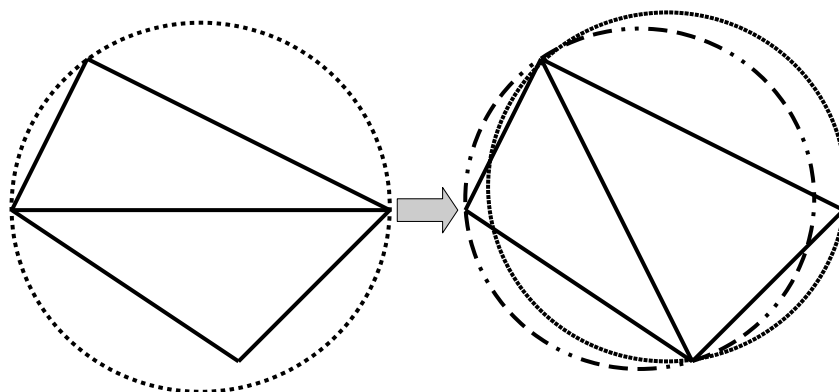


Рисунок 1.81 – Операція «flip» для трикутників

Для чотирикутників можна запропонувати аналогічну операцію зміни діагоналі (рис. 1.82). Для двох сусідніх чотирикутників існує два можливих

варіанти зміни діагоналі. Отже, порівнюючи максимальні вузли у вихідній конфігурації з двох альтернативних необхідно обрати той, що з найменшим максимальним кутом (або з найбільшим мінімальним). Водночас необхідною умовою застосування цієї операції є опуклість фігури, утвореної вершинами чотирикутників.

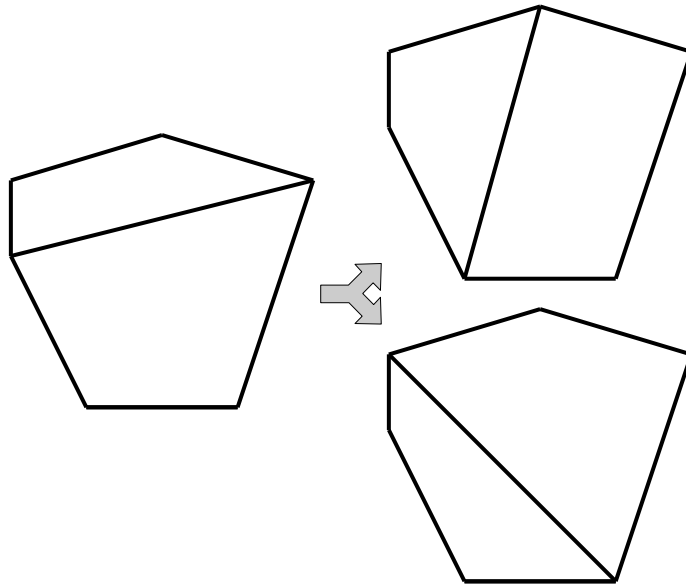


Рисунок 1.82 – Операція «flip» для чотирикутників

У обох випадках алгоритм полягає у послідовній перевірці необхідності застосування операції «flip» та її виконанні за необхідністю. Елементи необхідно перевіряти на відповідність критерію поки були випадки зміни діагоналі (критерієм зупинки є відсутність перебудов на елементах за ітерацію).

Підсумковий алгоритм застосування операції «flip» можна подати у такому вигляді:

algorithm flip

input:

M – дискретна модель межі об'єкта

output:

M – дискретна модель межі об'єкта

begin

do

$f \leftarrow \text{false}$

for each $e \in M.E$ **do**

for $i \leftarrow 1$ **to** $|e|$ **do**

$k_e \leftarrow (M.\text{incident}(e[i]) \cap M.\text{incident}(e[i+1])) \setminus M.\text{code}(e)$

if $\text{is\_flip}(M.\text{code}(e), k_e)$ **then**

$M.\text{flip}(M.\text{code}(e), k_e)$

```

    f ← true
  end if
end for
end for
while f = true
end

```

На рис. 1.83 порівнюються результати 8 ітерацій згладжування дискретної моделі межі об'єкта, заданого формулою (1.58), без використання процедури «flip» і з її використанням. Мінімальний кут в трикутних елементах дискретної моделі зростає з $7,83^\circ$ до $13,83^\circ$. Для чотирикутних елементів максимальний кут зменшується з $178,1^\circ$ до $174,89^\circ$.

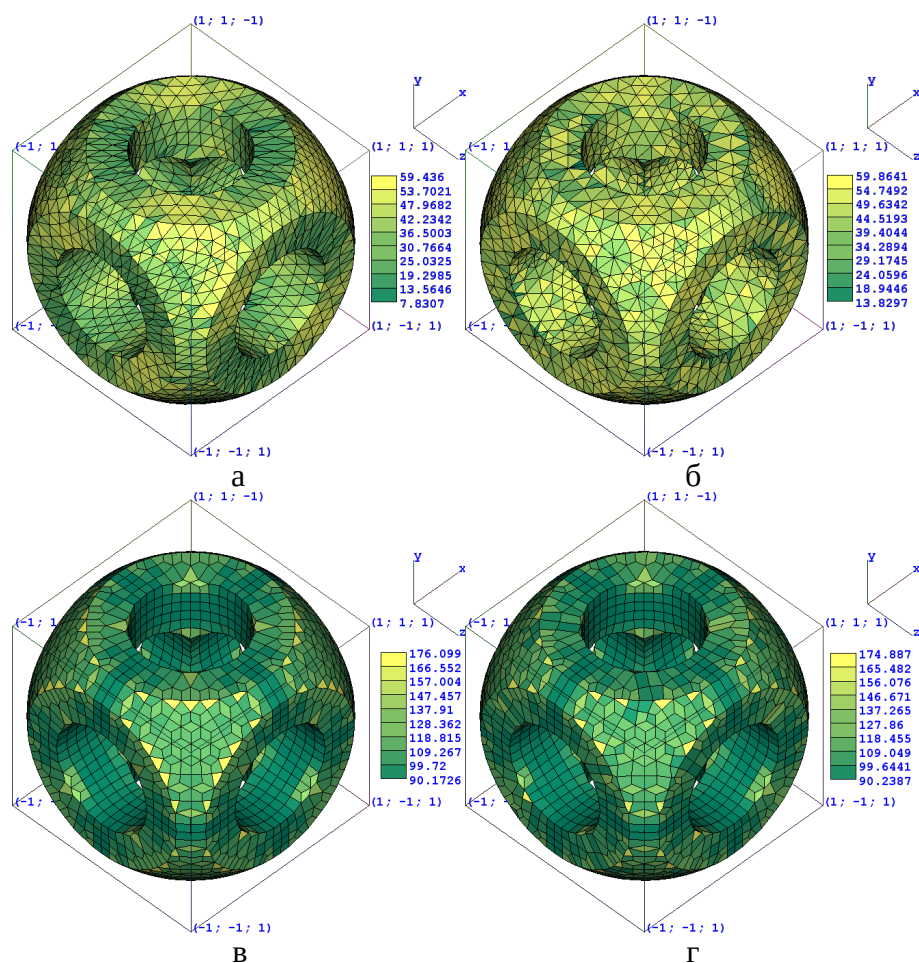


Рисунок 1.83 – Порівняння результатів застосування операції «flip»: а – результат згладжування для трикутників без використання операції «flip»; б – результат згладжування для трикутників з використанням операції «flip»; в – результат згладжування для чотирикутників без використання операції «flip»; г – результат згладжування для чотирикутників з використанням операції «flip»

Для моделі пробки, заданої формулою (1.33), на рис. 1.84 порівняно результати оптимізації відповідної дискретної моделі межі.

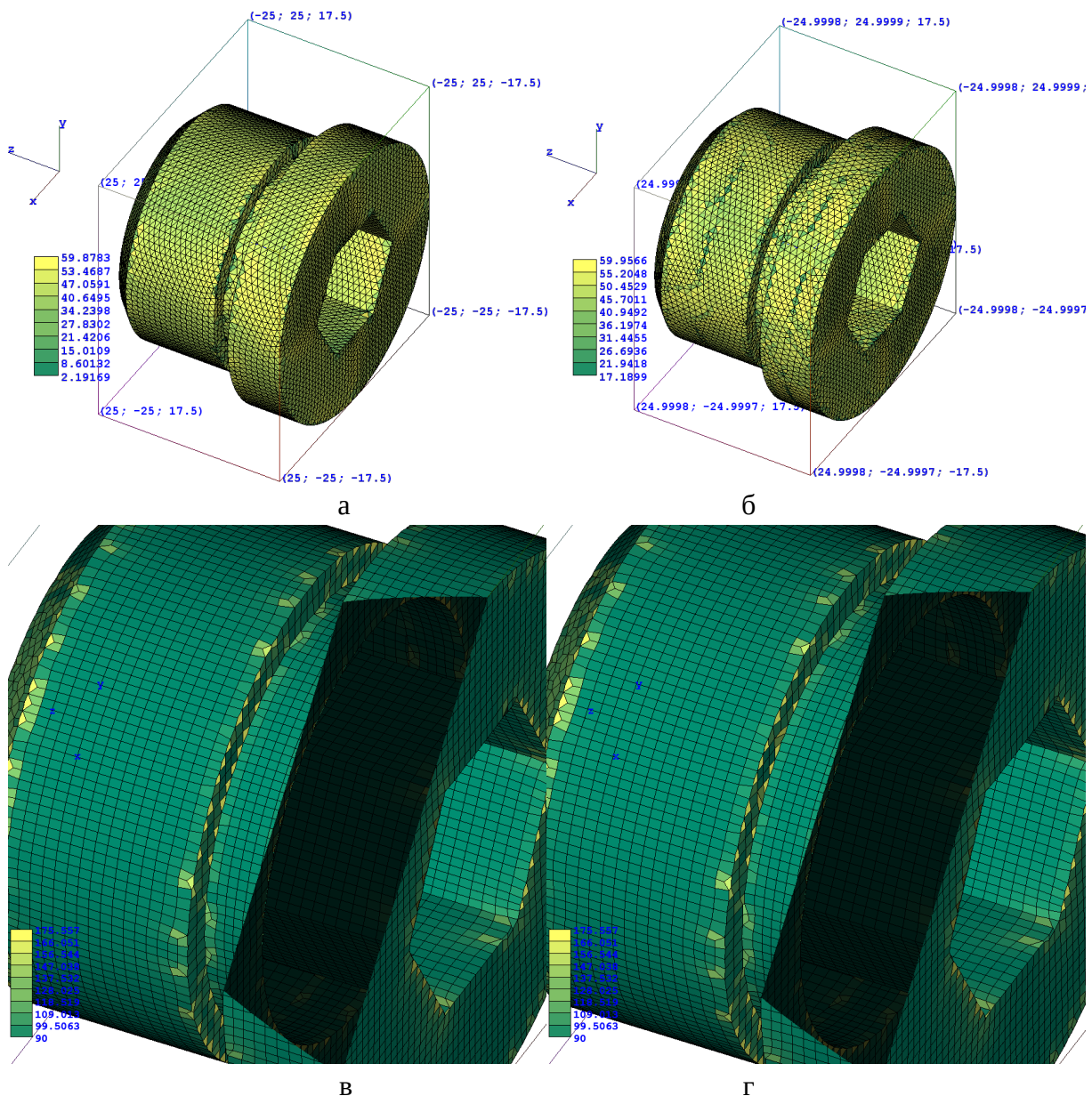


Рисунок 1.84 – Порівняння результатів застосування операції «flip»: а – результат згладжування для трикутників без використання операції «flip»; б – результат згладжування для трикутників з використанням операції «flip»; в – результат згладжування для чотирикутників без використання операції «flip»; г – результат згладжування для чотирикутників з використанням операції «flip»

Найбільш гострий кут у дискретній моделі межі пробки, заснований на трикутниках, якщо не використовувати операцію «flip», дорівнює $2,19^\circ$ (рис. 1.84, а). Якщо «flip» була використана, то у цій моделі мінімальний кут дорівнює $17,19^\circ$ (рис. 1.84, б). У дискретній моделі межі пробки на базі чотирикутників максимальний кут не змінився і дорівнює $175,56^\circ$ (рис. 1.84, в–г), що обумовлено апроксимацією зламів межі парою суміжних ребер.

Аналіз отриманих результатів дозволяє зробити висновок, що застосування операції «flip» (яка не вимагає вставки додаткових вузлів) дозволяє поліпшити геометричні характеристики дискретної моделі. Водночас

результат її застосування для трикутників більш помітний, ніж для чотирикутників. Це обумовлено тим, що вимога утворення опуклої фігури шістьма вузлами істотно знижує можливості її практичного застосування.

При використанні чотирикутників можливим способом зменшення великих кутів в елементах є розбиття їх на трикутники (проведенням ребра через найбільший кут) з подальшим перетворенням шляхом серединного розбиття. В такому випадку, після розбиття всіх необхідних чотирикутників на трикутники, формується нова множина чотирикутників шляхом з'єднання кожного вузла з центрами сусідніх у ньому сторін і геометричним центром елемента (рис. 1.85). Водночас для нових вузлів потрібно знайти їх проекції на межу об'єкта.

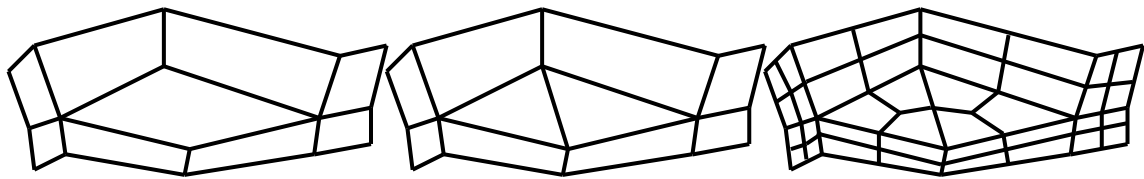


Рисунок 1.85 – Розбиття чотирикутників

Отже, розбиття елементів дозволяє зменшити значення максимального кута для дискретних моделей меж об'єктів на базі чотирикутників. Наприклад, для дискретної моделі межі «кубика», заданого формулою (1.58), (рис. 1.83, г) значення максимального кута в елементі зменшується з початкових $174,89^\circ$ до $155,13^\circ$ (рис. 1.86). Водночас зростає кількість вузлів і елементів. Наприклад, у вихідній дискретній моделі межі «кубика» було 3064 вузли та 3072 елементи, а у результаті розбиття – 13782 вузли та 13790 елементів.

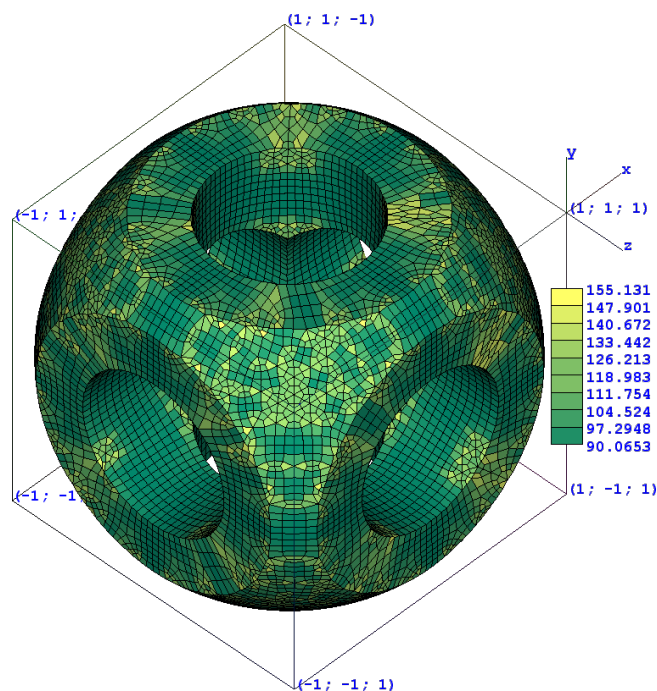


Рисунок 1.86 – Приклад використання розбиття чотирикутників

1.11. Дискретні моделі форм об'єктів на базі солідів

Описані вище методи генерації дискретних моделей меж об'єктів можна використовувати для розв'язання задач комп'ютерної графіки та чисельного дослідження поведінки оболонок. Проте, у практиці інженерного аналізу часто виникає необхідність використання дискретних моделей, які апроксимують всю геометричну область, а не тільки її межу. У такому випадку використовують моделі на базі солідів: трикутників або чотирикутників на площині та тетраедрів або шестигранників у просторі. Водночас дискретні моделі меж об'єктів можуть бути вихідними даними для методів генерації дискретних моделей на базі солідів.

1.11.1 Дво- і тривимірна тріангуляція

Для трикутників і тетраедрів, які спільно називають симплексами (симплекс – це фігура, яка є n -вимірним узагальненням трикутника), розробляють спільні методи побудови дискретних моделей форм об'єктів. Зазвичай ці методи поєднують під спільною назвою «тріангуляція».

Як зазначено раніше для множини точок на площині та планарних графів розроблені ефективні методи побудови тріангуляції на основі критерію Делоне [144]. Вважають, що дискретна модель відповідає цьому критерію, якщо у коло, описане навколо довільного трикутника, потрапляють тільки вузли, які належать цьому трикутнику.

Дискретна модель форми об'єкта на базі трикутників, яка відповідає критерію Делоне, має такі властивості [151]:

- єдиність, якщо довільні чотири точки не знаходяться на одному колі;
- максимальне значення мінімального кута в елементах дискретної моделі, які побудовані на даній множині точок;
- максимальна сума радіусів описаних кіл для трикутників, побудованих на даній множині точок.

У тривимірному випадку, коли елементами є тетраедри, а для перевірки критерію Делоне використовують описані навколо них сфери, властивість максимізації мінімального кута в елементах не зберігається. Проте, зважена за об'ємами інцидентних елементів сума квадратів довжин сторін буде мінімальною для заданої множини точок тільки при виконанні критерію Делоне [250].

1.11.1.1 Алгоритм метода тріангуляції

В обох випадках для побудови тріангуляції Делоне можна скористатися так званою суперобластю. Суперобласть – це елементарний опуклий багатогранник з трикутними гранями, який повністю включає всі точки межі форми об'єкта. У двовимірному випадку супер-областю може бути трикутник або квадрат, а в тривимірному – тетраедр або куб, грані якого розбиті на трикутники. Далі з'єднавши границю суперобласті з першим вузлом із дискретної моделі межі об'єкта отримаємо початкову тріангуляцію Делоне. Кожен наступний вузол із дискретної моделі межі об'єкта вставляється у тріангуляцію Делоне шляхом видалення всіх симплексів, в описаній сфері яких він потрапляє, та з'єднанням з гранями отриманого після видалення багатогранника. Після видалення елементів, які з'єднані з вершинами супер області, буде отримана тріангуляція Делоне для опуклого многовиду, утвореного вузлами дискретної моделі межі об'єкта. Однак, межа об'єкта може бути неопуклою. У такому випадку в тріангуляцію потрібно додавати проекції середини грані або ребра елемента на межу об'єкта, якщо ця грань або ребро поза власних вершин перетинається межею. Отже, для побудови дискретної моделі вигляду (1.57) для форми об'єкта, яка задана функціонально, можна записати такий алгоритм генерації тріангуляції Делоне:

algorithm delaunay

input:

F – функція, яка подає форму об'єкта Ω

B – дискретна модель межі об'єкта Ω

ϵ – чисельний нуль

output:

T – дискретна модель форми об'єкта Ω на базі симплексів

begin

$T \leftarrow \text{initial\_triangulation}(B)$

for each $n \in B.N$ **do**

$\text{insert\_delaunay}(T, n, \epsilon)$

end for

do

$\text{is\_subdivided} \leftarrow \text{false}$

for each $e \in T.E$ **do**

for $i \leftarrow 1$ **to** $|e|$ **do**

for each $b \in B.E$ **do**

if $\text{is\_crossed}(B.N(b), e[i], e[i+1], \epsilon) = \text{true}$ **then**

```


$$C \leftarrow \frac{1}{|b|} \sum_{j \in b} \mathbf{B}[j]$$


$$h \leftarrow \min_{(j \neq k) \in b} \|\mathbf{B}[j] - \mathbf{B}[k]\|$$

insert_delaunay(T, find_border(C, F, h,  $\epsilon$ ),  $\epsilon$ )
is_subdivided  $\leftarrow$  true
end if
end for
end for
end for
while is_subdivided = true
for each  $e \in \mathbf{T.E}$  do

$$C \leftarrow \frac{1}{|e|} \sum_{j \in e} \mathbf{T}[j]$$

if  $F(C) < 0$  then
remove_element(T, T.code(e))
end if
end for
return T

```

Алгоритм побудови початкової триангуляції initial_triangulation потрібно реалізувати з урахуванням розмірності простору. У двовимірному випадку як супер-областю можна скористатися кубом, який повністю охоплює всі вузли моделі межі області. У результаті початкова триангуляція буде містити чотири трикутники.

algorithm initial_triangulation

input:

B – дискретна модель межі об'єкта Ω

output:

T – початкова триангуляція

begin

$d \leftarrow 2 \max_{(n \neq m) \in \mathbf{B.N}} \|n - m\|$

$$C \leftarrow \frac{1}{|\mathbf{B.N}|} \sum_{i=1}^{|\mathbf{B.N}|} \mathbf{B}[i]$$

T $\leftarrow (\emptyset, \emptyset)$

$P_1 \leftarrow C + (-d; -d)$

$P_2 \leftarrow C + (d; -d)$

$P_3 \leftarrow C + (d; d)$

$P_4 \leftarrow C + (-d; d)$

insert_element(**T**, $\{P_1, P_2, \mathbf{B}[1]\}$)

insert_element(**T**, $\{P_2, P_3, \mathbf{B}[1]\}$)

insert_element(**T**, $\{P_3, P_4, \mathbf{B}[1]\}$)

```

insert_element(T, { $P_4$ ,  $P_1$ , B[1]})
return T

```

Наприклад, для дискретної моделі межі пластини (рис. 1.87, а), яка задана формулою (1.18), початкова тріангуляція прийме вигляд, наведений на рис. 1.87, б.

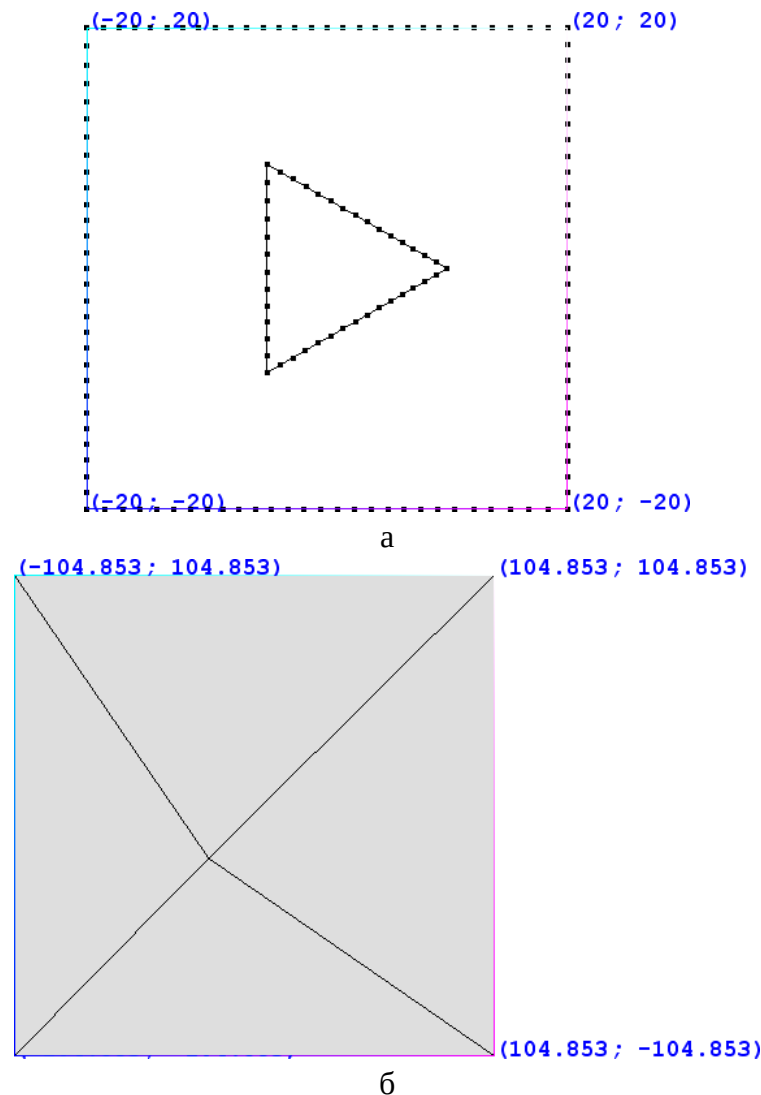


Рисунок 1.87 – Початкова тріангуляція: а – вихідна дискретна модель межі об'єкта;
б – початкова тріангуляція

Якщо у тривимірному випадку супер-областю буде куб, початкова тріангуляція буде мати 12 тетраедрів (по два на кожну грань куба). Отже, алгоритм `initial_triangulation` можна записати у такому вигляді:

```

algorithm initial_triangulation
input:
    B – дискретна модель межі об'єкта  $\Omega$ 
output:

```

T – початкова триангуляція

begin

$$d \leftarrow 2 \max_{(n \neq m) \in \mathbf{B} \cdot \mathbf{N}} \|n - m\|$$

$$C \leftarrow \frac{1}{|\mathbf{B} \cdot \mathbf{N}|} \sum_{i=1}^{|\mathbf{B} \cdot \mathbf{N}|} \mathbf{B}[i]$$

$$\mathbf{T} \leftarrow (\emptyset, \emptyset)$$

$$P_1 \leftarrow C + (-d; -d; -d)$$

$$P_2 \leftarrow C + (d; -d; -d)$$

$$P_3 \leftarrow C + (d; d; -d)$$

$$P_4 \leftarrow C + (-d; d; -d)$$

$$P_5 \leftarrow C + (-d; -d; d)$$

$$P_6 \leftarrow C + (d; -d; d)$$

$$P_7 \leftarrow C + (d; d; d)$$

$$P_8 \leftarrow C + (-d; d; d)$$

insert_element(**T**, { $P_1, P_2, P_3, \mathbf{B}[1]$ })

insert_element(**T**, { $P_1, P_3, P_4, \mathbf{B}[1]$ })

insert_element(**T**, { $P_4, P_3, P_8, \mathbf{B}[1]$ })

insert_element(**T**, { $P_8, P_3, P_7, \mathbf{B}[1]$ })

insert_element(**T**, { $P_5, P_7, P_6, \mathbf{B}[1]$ })

insert_element(**T**, { $P_5, P_8, P_7, \mathbf{B}[1]$ })

insert_element(**T**, { $P_1, P_4, P_8, \mathbf{B}[1]$ })

insert_element(**T**, { $P_1, P_8, P_5, \mathbf{B}[1]$ })

insert_element(**T**, { $P_2, P_6, P_3, \mathbf{B}[1]$ })

insert_element(**T**, { $P_3, P_6, P_7, \mathbf{B}[1]$ })

insert_element(**T**, { $P_8, P_3, P_7, \mathbf{B}[1]$ })

insert_element(**T**, { $P_1, P_5, P_2, \mathbf{B}[1]$ })

insert_element(**T**, { $P_2, P_5, P_6, \mathbf{B}[1]$ })

return T

Алгоритм insert_delaunay(**T**, n , ϵ) додає вузол n у дискретну модель **T**. У цьому алгоритмі, після перевірки існування такого вузла у **T** видаляються всі елементи, в описане коло (або сферу тривимірному випадку) яких потрапляє вузол n . У результаті утворюється багатокутник (багатогранник), з'єднання ребер (граней) якого з n дозволяє утворити множину елементів. Алгоритм можна формалізувати у такому вигляді:

algorithm insert_delaunay

input:

T – триангуляція

n – вузол

ϵ – чисельний нуль

output:

T – триангуляція

```

begin
   $is\_exist \leftarrow false$ 
  for each  $n_t \in T.N$  do
    if  $\|n.v - n_t.v\| < \varepsilon$  then
       $is\_exist \leftarrow true$ 
    end if
  end for
  if  $is\_exist = false$  then
     $L \leftarrow (\emptyset, \emptyset)$ 
    for each  $e \in T.E$  do
       $(C, r) \leftarrow circum\_circle(T.N(e))$ 
      if  $\|C - n.v\| < r$  then
         $insert\_element(L, T.N(e), \varepsilon)$ 
         $remove\_element(T, T.code(e))$ 
      end if
    end for
     $B \leftarrow \delta(L)$ 
    for each  $e \in B$  do
       $insert\_element(T, \{B.N(e), n\}, \varepsilon)$ 
    end for
  end if
end

```

Алгоритм `circum_circle` необхідний для перевірки потрапляння точки в описане коло або сферу. Для цього необхідно знайти координати її центру та радіус. При розв'язанні цієї задачі у двовимірному випадку необхідно обчислити значення чотирьох визначників матриць 3×3 [305]. У результаті алгоритм можна подати у такому вигляді:

algorithm `circum_circle`

input:

V – список координат вершин двовимірного трикутного елемента

output:

C – координати центра описаного навколо V кола

r – радіус кола, описаного навколо V

begin

$$a \leftarrow \begin{vmatrix} V[1].x & V[1].y & 1 \\ V[2].x & V[2].y & 1 \\ V[3].x & V[3].y & 1 \end{vmatrix}$$

$$b_x \leftarrow - \begin{vmatrix} V[1].x^2 + V[1].y^2 & V[1].y & 1 \\ V[2].x^2 + V[2].y^2 & V[2].y & 1 \\ V[3].x^2 + V[3].y^2 & V[3].y & 1 \end{vmatrix}$$

$$b_y \leftarrow \begin{vmatrix} \mathbf{V}[1].x^2 + \mathbf{V}[1].y^2 & \mathbf{V}[1].x & 1 \\ \mathbf{V}[2].x^2 + \mathbf{V}[2].y^2 & \mathbf{V}[2].x & 1 \\ \mathbf{V}[3].x^2 + \mathbf{V}[3].y^2 & \mathbf{V}[3].x & 1 \end{vmatrix}$$

$$c \leftarrow - \begin{vmatrix} \mathbf{V}[1].x^2 + \mathbf{V}[1].y^2 & \mathbf{V}[1].x & \mathbf{V}[1].y \\ \mathbf{V}[2].x^2 + \mathbf{V}[2].y^2 & \mathbf{V}[2].x & \mathbf{V}[2].y \\ \mathbf{V}[3].x^2 + \mathbf{V}[3].y^2 & \mathbf{V}[3].x & \mathbf{V}[3].y \end{vmatrix}$$

$$C \leftarrow \left(-\frac{b_x}{2a}; -\frac{b_y}{2a} \right)$$

$$r \leftarrow \frac{\sqrt{b_x^2 + b_y^2 - 4ac}}{2|a|}$$

return (C, r)

У тривимірному випадку для знаходження центру та радіусу описаної навколо тетраедра сфери необхідно знайти значення вже шести визначників матриць 4×4, а алгоритм `circum_circle` матиме такий вигляд:

algorithm `circum_circle`

input:

V – список координат вершин тетраедра

output:

C – координати центра описаної навколо **V** сфери

r – радіус описаної навколо **V** сфери

begin

$$a \leftarrow \begin{vmatrix} \mathbf{V}[1].x & \mathbf{V}[1].y & \mathbf{V}[1].z & 1 \\ \mathbf{V}[2].x & \mathbf{V}[2].y & \mathbf{V}[2].z & 1 \\ \mathbf{V}[3].x & \mathbf{V}[3].y & \mathbf{V}[3].z & 1 \\ \mathbf{V}[4].x & \mathbf{V}[4].y & \mathbf{V}[4].z & 1 \end{vmatrix}$$

$$d_x \leftarrow \begin{vmatrix} \mathbf{V}[1].x^2 + \mathbf{V}[1].y^2 + \mathbf{V}[1].z^2 & \mathbf{V}[1].y & \mathbf{V}[1].z & 1 \\ \mathbf{V}[2].x^2 + \mathbf{V}[2].y^2 + \mathbf{V}[2].z^2 & \mathbf{V}[2].y & \mathbf{V}[2].z & 1 \\ \mathbf{V}[3].x^2 + \mathbf{V}[3].y^2 + \mathbf{V}[3].z^2 & \mathbf{V}[3].y & \mathbf{V}[3].z & 1 \\ \mathbf{V}[4].x^2 + \mathbf{V}[4].y^2 + \mathbf{V}[4].z^2 & \mathbf{V}[4].y & \mathbf{V}[4].z & 1 \end{vmatrix}$$

$$d_y \leftarrow \begin{vmatrix} \mathbf{V}[1].x^2 + \mathbf{V}[1].y^2 + \mathbf{V}[1].z^2 & \mathbf{V}[1].z & \mathbf{V}[1].x & 1 \\ \mathbf{V}[2].x^2 + \mathbf{V}[2].y^2 + \mathbf{V}[2].z^2 & \mathbf{V}[2].z & \mathbf{V}[2].x & 1 \\ \mathbf{V}[3].x^2 + \mathbf{V}[3].y^2 + \mathbf{V}[3].z^2 & \mathbf{V}[3].z & \mathbf{V}[3].x & 1 \\ \mathbf{V}[4].x^2 + \mathbf{V}[4].y^2 + \mathbf{V}[4].z^2 & \mathbf{V}[4].z & \mathbf{V}[4].x & 1 \end{vmatrix}$$

$$d_z \leftarrow \begin{vmatrix} \mathbf{V}[1].x^2 + \mathbf{V}[1].y^2 + \mathbf{V}[1].z^2 & \mathbf{V}[1].x & \mathbf{V}[1].y & 1 \\ \mathbf{V}[2].x^2 + \mathbf{V}[2].y^2 + \mathbf{V}[2].z^2 & \mathbf{V}[2].x & \mathbf{V}[2].y & 1 \\ \mathbf{V}[3].x^2 + \mathbf{V}[3].y^2 + \mathbf{V}[3].z^2 & \mathbf{V}[3].x & \mathbf{V}[3].y & 1 \\ \mathbf{V}[4].x^2 + \mathbf{V}[4].y^2 + \mathbf{V}[4].z^2 & \mathbf{V}[4].x & \mathbf{V}[4].y & 1 \end{vmatrix}$$

$$c \leftarrow \begin{vmatrix} \mathbf{V}[1].x^2 + \mathbf{V}[1].y^2 + \mathbf{V}[1].z^2 & \mathbf{V}[1].x & \mathbf{V}[1].y & \mathbf{V}[1].z \\ \mathbf{V}[2].x^2 + \mathbf{V}[2].y^2 + \mathbf{V}[2].z^2 & \mathbf{V}[2].x & \mathbf{V}[2].y & \mathbf{V}[2].z \\ \mathbf{V}[3].x^2 + \mathbf{V}[3].y^2 + \mathbf{V}[3].z^2 & \mathbf{V}[3].x & \mathbf{V}[3].y & \mathbf{V}[3].z \\ \mathbf{V}[4].x^2 + \mathbf{V}[4].y^2 + \mathbf{V}[4].z^2 & \mathbf{V}[4].x & \mathbf{V}[4].y & \mathbf{V}[4].z \end{vmatrix}$$

$$C \leftarrow \left(\frac{d_x}{2a}, \frac{d_y}{2a}, \frac{d_z}{2a} \right)$$

$$r \leftarrow \frac{\sqrt{d_x^2 + d_y^2 + d_z^2 - 4ac}}{2|a|}$$

return (C, r)

Алгоритм `is_crossed` використовується для перевірки перетину відрізком елемента дискретної моделі межі області. У двовимірному випадку цей алгоритм має такий вигляд:

algorithm `is_crossed`

input:

V – пара двовимірних точок, які відповідають елементу межі двовимірної області

P_1 – перша точка відрізка

P_2 – друга точка відрізка

ε – чисельний нуль

output:

true, якщо (P_1, P_2) перетинає **V**, false – інакше

begin

p $\leftarrow P_1$

r $\leftarrow P_2 - P_1$

q $\leftarrow \mathbf{V}[1]$

s $\leftarrow \mathbf{V}[2] - \mathbf{V}[1]$

d $\leftarrow \mathbf{r} \times \mathbf{s}$

if $|d| < \varepsilon$ **then**

return false

end if

t $\leftarrow \frac{(\mathbf{q} - \mathbf{p}) \times \mathbf{s}}{d}$

u $\leftarrow \frac{(\mathbf{q} - \mathbf{p}) \times \mathbf{r}}{d}$

if $\varepsilon \leq t$ **and** $t \leq 1 - \varepsilon$ **and** $\varepsilon \leq u$ **and** $u \leq 1 - \varepsilon$ **then**

return true

end if

return false

У тривимірному випадку алгоритм `is_crossed` ґрунтується на перевірці перетину трикутника відрізком [306]. Зазначений алгоритм можна подати у такому вигляді:

algorithm `is_crossed`

input:

V – список тривимірних точок, які відповідають трикутному елементу

P_1 – перша точка відрізка

P_2 – друга точка відрізка

ε – чисельний нуль

output:

true, якщо (P_1, P_2) перетинає V , false – інакше

begin

$o \leftarrow P_1$

$d \leftarrow P_2 - P_1$

$e_1 \leftarrow V[2] - V[1]$

$e_2 \leftarrow V[3] - V[1]$

$t \leftarrow o - V[1]$

$p \leftarrow d \times e_2$

$q \leftarrow t \times e_1$

$$\begin{bmatrix} \tau \\ u \\ v \end{bmatrix} = \frac{1}{p \cdot e_1} \begin{bmatrix} q \cdot e_2 \\ p \cdot t \\ q \cdot d \end{bmatrix}$$

if $\tau < \varepsilon$ **or** $1 - \varepsilon < \tau$ **or** $u + v > 1 - \varepsilon$ **or** $u < \varepsilon$ **or** $v < \varepsilon$ **then**

return false

end if

return true

Останнім кроком алгоритму `delaunay` є видалення зовнішніх елементів. Для цього із результатів вставки вузлів у тріангуляцію Делоне для суперобласті видаляються всі елементи, у середині яких функція області менша нуля (наприклад, супер-область – рис. 1.88, а, результат видалення зовнішніх елементів – рис. 1.88, б).

Необхідно відзначити, що у тривимірному випадку існують дискретні моделі меж геометричних об'єктів, для яких неможлива побудова множини тетраедрів без додавання нових вузлів [285]. Також у [308] доведено, що для вихідної дискретної моделі межі об'єкта перевірка існування множини тетраедрів, яка задовольняє критерію Делоне, є NP-повною.

У [156] доведена можливість побудови множини тетраедрів, що задовольняють критерію Делоне, якщо для кожного ребра вихідної тріангуляції не існує вузлів, що належать цього ребра та потрапляють у сферу, діаметром якої є це ребро.

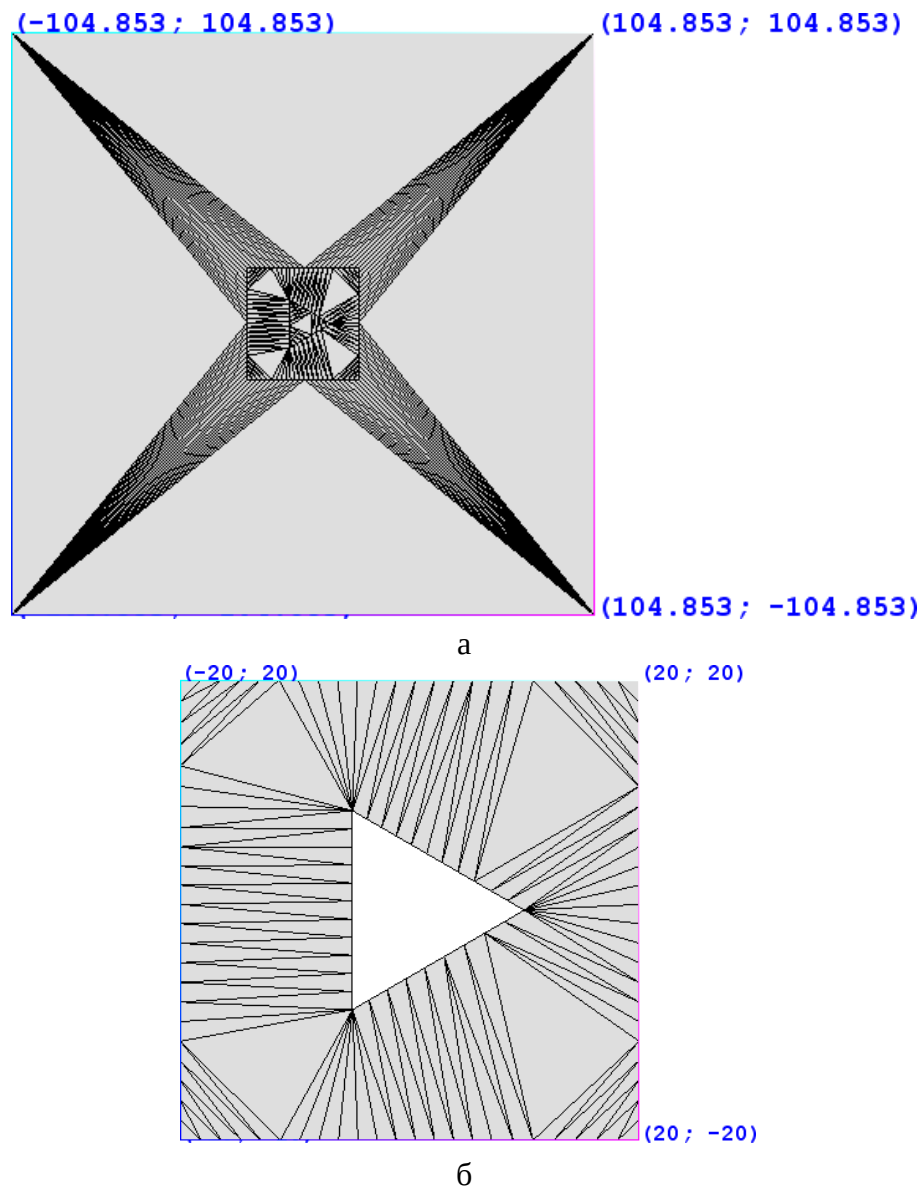


Рисунок 1.88 – Триангуляція Делоне: а – побудована для суперобласті; б – результат видалення зовнішніх елементів для моделі пластинки

Алгоритм *delaunay* послідовно будує множину тетраедрів з перевіркою умови Делоне при вставці. Для забезпечення апроксимації меж об'єкта ребрами та гранями тетраедрів нові вузли додаються у точки перетину ребрами меж (якщо такі відрізняються від кінців ребра).

1.11.1.2 Алгоритм оптимізації чисельних характеристик дискретної моделі

З точки зору інженерного аналізу об'єктів машинобудування важливими є чисельні характеристики дискретних моделей форм. Наявність значних

диспропорцій у елементах (дуже гострих кутів) зумовлює малі значення якобіана при скінченно-елементному аналізі з використанням місцевих координат. У роботі [162] доведено, що вставкою вузлів у центри трикутників, які не відповідають вимозі відносно гострого кута, і у середини граничних ребер, у діаметральні кола яких потрапляють вузли, що їм не належать, найбільш гострий кут буде не менший за $20,7^\circ$ (співвідношення сторін елементів буде не більше $\sqrt{2}$). Проте, цю техніку не можна застосовувати для тетраедрів, тому необхідно використовувати додаткове розбиття межових граней зі вставкою нового вузла у центри їх описаних кіл [309].

У результаті узагальнення методів, запропонованих у [162] і [309], на випадок використання дискретної моделі вигляду (1.57) при функціональному поданні форми об'єкта можна сформулювати такий алгоритм:

algorithm refine_delaunay

input:

D – дво- або тривимірна тріангуляція Делоне форми об'єкта Ω

F – функція, яка подає форму об'єкта Ω

α – обмеження на відношення довжин сторін елемента

ε – чисельний нуль

output:

D – дво- або тривимірна тріангуляція Делоне форми об'єкта Ω

begin

B $\leftarrow \delta(\mathbf{D})$

S $\leftarrow \text{encroached}(\mathbf{D}, \varepsilon) \cup \text{poor}(\mathbf{D}, \alpha)$

while **S** $\neq \emptyset$ **do**

if **s** $\in \mathbf{S}$ – сегмент **then**

h $\leftarrow 0,5 \|\mathbf{B}[s[1]] - \mathbf{B}[s[2]]\|$

C $\leftarrow \text{find\_border}(0,5 (\mathbf{B}[s[1]] + \mathbf{B}[s[2]]), \mathbf{F}, h, \varepsilon)$

 insert_delaunay(**D**, **C**, ε)

else if **s** $\in \mathbf{S}$ – тривимірний трикутник **then**

C $\leftarrow \text{circum\_circle}(\mathbf{D.N}(s))$

 insert_delaunay(**D**, **C**, ε)

else if **s** $\in \mathbf{S}$ – двовимірний трикутник або тетраедр

C $\leftarrow \text{circum\_circle}(\mathbf{D.N}(s))$

f $\leftarrow \text{false}$

B $\leftarrow \delta(\mathbf{D})$

if $\exists e \in \mathbf{B}: \text{is\_encroach}(\mathbf{B.N}(e), \mathbf{C}, \varepsilon) = \text{true}$ **then**

S $\leftarrow \mathbf{S} \cup e$

else

 insert_delaunay(**D**, **C**, ε)

end if

end if

end while

return D

На першому етапі алгоритму формується множина, яка складається з сегментів, граней та елементів, що підлягають розбиттю. Для цього використано два алгоритми: *encroached* і *poor*. Алгоритм *encroached* формує список ребер і граней (у тривимірному випадку). Для двовимірних дискретних моделей форм об'єктів його можна подати у такому вигляді:

algorithm encroached

input:

D – двовимірна тріангуляція форми об'єкта Ω

ϵ – чисельний нуль

output:

S – множина межових ребер, які потрібно розбити

begin

B $\leftarrow \delta(D)$

S $\leftarrow \emptyset$

for each $b \in B$ **do**

if $\exists i \notin b: \text{is\_encroach}(\{D[b[1]], D[b[2]]\}, D[i], \epsilon) = \text{true}$ **then**

S $\leftarrow S \cup b$

end if

end for

return S

У тривимірному випадку результат алгоритму *encroached* повинен включати ребра та грані, які необхідно розбити. У цьому випадку алгоритм прийме такий вигляд:

algorithm encroached

input:

D – тривимірна тріангуляція форми об'єкта Ω

ϵ – чисельний нуль

output:

S – множина межових ребер і граней, які потрібно розбити

begin

B $\leftarrow \delta(D)$

S $\leftarrow \emptyset$

for each $b \in B$ **do**

if $\exists i \notin b: \text{is\_encroach}(\{D[b[1]], D[b[2]]\}, D[i], \epsilon) = \text{true}$ **then**

S $\leftarrow S \cup (b[1], b[2])$

else if $\exists i \notin b: \text{is\_encroach}(\{D[b[2]], D[b[3]]\}, D[i], \epsilon) = \text{true}$ **then**

S $\leftarrow S \cup (b[2], b[3])$

```

else if  $\exists i \notin b$ : is_encroach( $\{\mathbf{D}[b[3]], \mathbf{D}[b[1]]\}$ ,  $\mathbf{D}[i]$ ,  $\varepsilon$ ) = true then
     $S \leftarrow S \cup (b[3], b[1])$ 
end if
end for
return  $S$ 

```

Алгоритм перевірки необхідності розбиття is_encroach матиме дві реалізації: для ребер і трикутних граней. Для дво- і тривимірних ребер досить перевірити як співвідноситься відстань від центру ребра до точки з його довжиною. У результаті отримаємо такий алгоритм:

algorithm is_encroach

input:

V – список з координат двох вузлів, які належать одному ребру

P – координати точки для перевірки

ε – чисельний нуль

output:

true, якщо P потрапить у коло або сферу, діаметром якої є V , false – інакше

begin

$C \leftarrow 0,5 (V[1] + V[2])$

$r \leftarrow 0,5 \|V[2] - V[1]\|$

return $\|P - C\| < r - \varepsilon$

Для трикутних граней перевірку попадання точки в описану екваторіальну сферу можна виконати у місцевих координатах. Для цього необхідно знайти координати вершин трикутника та заданої точки у системі координат, площина xOy якої збігається з площиною елемента, а вісь Oz з його нормаллю. У результаті отримаємо такий алгоритм:

algorithm is_encroach

input:

V – список з координат трьох вузлів, які належать одній грані

P – координати точки для перевірки

ε – чисельний нуль

output:

true, якщо P потрапить у екваторіальну сферу, описану навколо V , false – інакше

begin

$\mathbf{ab} \leftarrow V[2] - V[1]$

$\mathbf{ac} \leftarrow V[3] - V[1]$

$\mathbf{n} \leftarrow \mathbf{ab} \times \mathbf{ac}$

```


$$\mathbf{v}_x \leftarrow \frac{\mathbf{ab}}{\|\mathbf{ab}\|}$$


$$\mathbf{v}_z \leftarrow \frac{\mathbf{n}}{\|\mathbf{n}\|}$$


$$\mathbf{v}_y \leftarrow \mathbf{v}_x \times \mathbf{v}_z$$


$$\lambda \leftarrow \begin{bmatrix} \mathbf{v}_{x,x} & \mathbf{v}_{x,y} & \mathbf{v}_{x,z} \\ \mathbf{v}_{y,x} & \mathbf{v}_{y,y} & \mathbf{v}_{y,z} \\ \mathbf{v}_{z,x} & \mathbf{v}_{z,y} & \mathbf{v}_{z,z} \end{bmatrix}$$


$$v_1 \leftarrow \lambda (\mathbf{V}[1] - \mathbf{V}[1])$$


$$v_2 \leftarrow \lambda (\mathbf{V}[2] - \mathbf{V}[1])$$


$$v_3 \leftarrow \lambda (\mathbf{V}[3] - \mathbf{V}[1])$$


$$p \leftarrow \lambda (P - \mathbf{V}[1])$$


$$(C, r) \leftarrow \text{circum\_circle}(\{v_1, v_2, v_3\})$$

return  $\|p - C\| < r - \varepsilon$ 

```

Пошук центру описаної екваторіальній сфери для трикутної грані (як в наведений вище процедурі) можна виконати у місцевій системі координат. Для цього спочатку знаходиться центр описаного кола у місцевій системі координат, а потім за допомогою оберненого перетворення координат отримуються його координати у вихідній системі координат. У результаті отримаємо такий алгоритм:

algorithm `circum_circle`

input:

$\mathbf{V}$ – список з координат трьох вузлів, які належать одній грані

output:

C – координати центра описаної навколо $\mathbf{V}$ екваторіальної сфери

r – радіус описаної навколо $\mathbf{V}$ екваторіальної сфери

begin

$\mathbf{ab} \leftarrow \mathbf{V}[2] - \mathbf{V}[1]$

$\mathbf{ac} \leftarrow \mathbf{V}[3] - \mathbf{V}[1]$

$\mathbf{n} \leftarrow \mathbf{ab} \times \mathbf{ac}$

$$\mathbf{v}_x \leftarrow \frac{\mathbf{ab}}{\|\mathbf{ab}\|}$$

$$\mathbf{v}_z \leftarrow \frac{\mathbf{n}}{\|\mathbf{n}\|}$$

$$\mathbf{v}_y \leftarrow \mathbf{v}_x \times \mathbf{v}_z$$

$$\lambda \leftarrow \begin{bmatrix} \mathbf{v}_{x,x} & \mathbf{v}_{x,y} & \mathbf{v}_{x,z} \\ \mathbf{v}_{y,x} & \mathbf{v}_{y,y} & \mathbf{v}_{y,z} \\ \mathbf{v}_{z,x} & \mathbf{v}_{z,y} & \mathbf{v}_{z,z} \end{bmatrix}$$

$$v_1 \leftarrow \lambda (\mathbf{V}[1] - \mathbf{V}[1])$$

$$v_2 \leftarrow \lambda (\mathbf{V}[2] - \mathbf{V}[1])$$

$$v_3 \leftarrow \lambda (\mathbf{V}[3] - \mathbf{V}[1])$$

```

 $p \leftarrow \lambda (P - \mathbf{V}[1])$ 
 $(C, r) \leftarrow \text{circum\_circle}(\{v_1, v_2, v_3\})|$ 
return  $(\mathbf{V}[1] + \lambda^T C, r)$ 

```

Алгоритм формування списку «поганих» елементів роор ґрунтується на аналізі відношення найбільш довгої сторони елемента до найбільш короткої. Зрозуміло, чим ближче це співвідношення до одиниці, тим ближче елемент за формою до правильного. Цій алгоритм можна записати у загальному вигляді для дискретних моделей форм об'єктів у такому вигляді:

algorithm poor

input:

D – дво- або тривимірна тріангуляція Делоне форми об'єкта Ω

α – обмеження на відношення довжин сторін елемента

output:

S – множина елементів, які необхідно розбити

begin

$S \leftarrow \emptyset$

for each $e \in D.E$ **do**

$l_{min} \leftarrow \min_{i \in [1; |e|]} \|D[i+1] - D[i]\|$

$l_{max} \leftarrow \max_{i \in [1; |e|]} \|D[i+1] - D[i]\|$

if $\frac{l_{max}}{l_{min}} > \alpha$ **then**

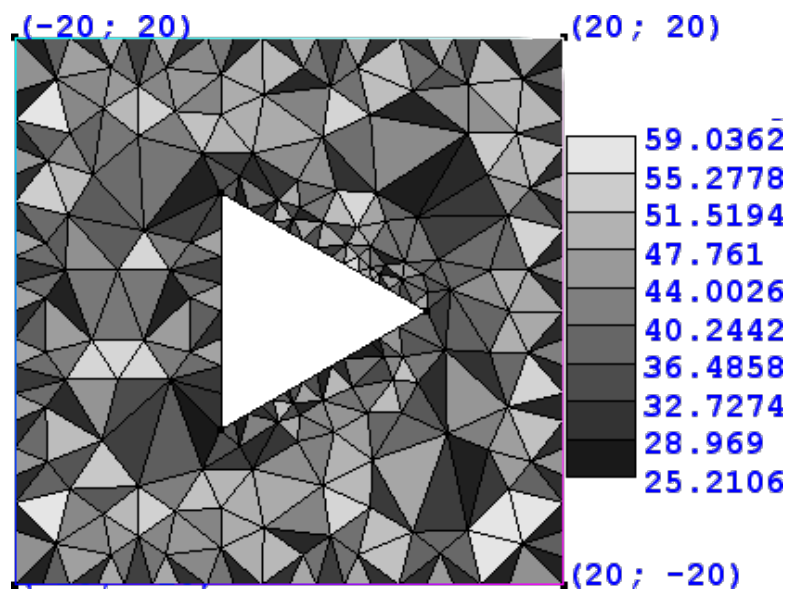
$S \leftarrow S \cup e$

end for

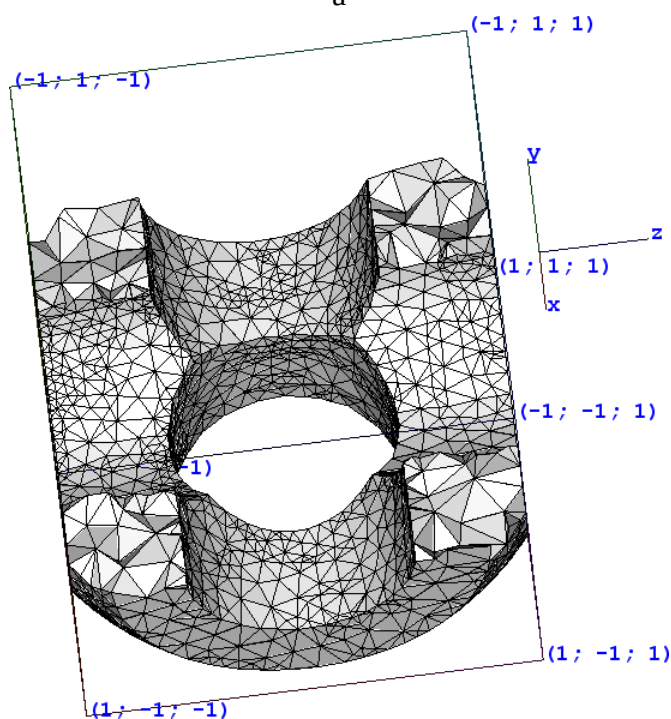
return S

У результаті застосування алгоритму покращення дво- або тривимірних тріангуляцій будуть усунені елементи, у яких співвідношення довжин сторін буде не нижче заданого (якщо задане значення вище теоретичної оцінки). Наприклад, на рис. 1.89, а зображено результат покращення тріангуляції Делоне для пластинки (рис. 1.76), якщо $\alpha=2$, а на рис. 1.89, б – переріз площиною $x=0$ результату покращення тріангуляції для моделі об'єкта, який заданий формулою (1.58).

Отже, запропоновані узагальнені методи дозволяють будувати дискретні моделі на базі дво- або тривимірних симплексів, що задовольняють критерію Делоне. У основі цих алгоритмів ідея вставки нових вузлів з одночасною перевіркою виконання умови Делоне. Проте, техніка вставки нового вузла практично не може бути застосована для дискретних моделей форм об'єктів на базі чотирикутників або шестигранників.



а



б

Рисунок 1.89 – Покращення триангуляції Делоне: а – двовимірна триангуляція; б – тривимірна триангуляція

1.11.2 Метод генерації дискретних моделей на базі елементів типу солід для форм об'єктів, заданих функціонально

Проекційний метод побудови дискретних моделей меж об'єктів можна використати як основу при розробці метода генерації дискретних моделей на базі елементів типу солід для форм об'єктів, заданих функціонально. В алгоритмі `background_grid_surface` з фонові дискретної моделі видаляються всі

зовнішні вузли, а також вузли у яких існує неоднозначна інтерпретація напрямку нормалі. Потім генерується модель межі шляхом обчислення проекцій межових граней елементів, що залишились, на межу об'єкта. Відповідно, для побудови дискретної моделі всього об'єкта потрібно заповнити елементами (солідами) шар простору, який залишився між елементами фонові дискретної моделі та елементами дискретної моделі межі об'єкта. Водночас необхідно враховувати, що між межовими гранями фонові дискретної моделі й елементами на межі існує взаємно-однозначна відповідність. Як результат можна сформулювати метод генерації дискретних моделей форм об'єктів, які задані функціонально, у вигляді такого алгоритму:

algorithm background-grid-mesh

input:

F – функція, яка подає форму об'єкта Ω

G – фонові дискретна модель

ε – чисельний нуль

output:

M – дискретна модель форми об'єкта Ω

begin

for each $n \in G.N$ **do**

if $F(n.v) < 0$ **then**

 remove_node($G, G.code(n)$)

end if

end for

$V_0 \leftarrow$ вузли дотику елементів G

while $V_0 \neq \emptyset$ **do**

$T \leftarrow \bigcup_{v \in V_0} G.incident(v)$

$k_{e0} \leftarrow \operatorname{argmin}_{k_e \in T} (G.power(G(k_e)))$

 remove_element(G, k_{e0})

$V_0 \leftarrow$ вузли дотику елементів G

end while

$B \leftarrow \delta(G)$

$M \leftarrow G$

$h \leftarrow \min_{(n_0 \neq n_1) \in M.N} \|n_0.v - n_1.v\|$

for each $e \in B.E$ **do**

$V \leftarrow \emptyset$

for $i \leftarrow |e|$ **down-to** 1 **do**

$V \leftarrow B.N[e(i)]$

end for

for $i \leftarrow 1$ **to** $|e|$ **do**


```

     $V \leftarrow V \cup \{\text{find\_border}(V[i], F, h, \varepsilon)\}$ 
  end for
  add_elements( $M, V, \varepsilon$ )
end for
return  $M$ 

```

На першому етапі алгоритму формується початкова дискретна модель шляхом видалення зовнішніх вузлів із фонові. Далі для усунення точок дотику з початкової моделі видаляються елементи, які суміжні у відповідних вузлах. Для кожної межевої грані початкової дискретної моделі обчислюється її проекція на межі об'єкта Ω . На заключному етапі формуються межові елементи. Для цього вузли межових граней та їх проекції об'єднуються у множину V , для якої будуються елементи за допомогою алгоритму `add_elements`. Множина V будується з урахуванням того, що порядок обходу вершин у спільній грані сусідніх елементів протилежний. Якщо елементами фонові дискретної моделі є чотирикутники або шестигранники, то для кожної межевої грані формується тільки один елемент. У такому випадку алгоритм `add_elements` отримає такий вигляд:

algorithm `add_elements`

input:

M – дискретна модель на базі двовимірних чотирикутників або шестигранників

V – список вершин

ε – чисельний нуль

begin

`insert_element`(M, V, ε)

end

Якщо елементами фонові дискретної моделі є трикутники, то для кожного межевого ребра початкової сітки можна двома способами побудувати елементи (змінюючи напрям діагоналі у чотирикутнику, який утворено суміжними межевими вершинами та їх проекціями). З двох можливих способів побудови трикутників у межевому шарі (рис. 1.90) раціонально обирати той, при якому сума довжин сторін буде мінімальною. Якщо вважати, що вершини збережені за порядком обходу супроти годинникової стрілки, то отримаємо такий алгоритм:

algorithm `add_elements`

input:

M – дискретна модель на базі двовимірних трикутників

V – список вершин

ε – чисельний нуль

begin

if $\|V[3] - V[1]\| < \|V[4] - V[2]\|$ **then**

 insert_element(M , $\{V[1], V[2], V[3]\}$, ε)

 insert_element(M , $\{V[1], V[3], V[4]\}$, ε)

else

 insert_element(M , $\{V[1], V[2], V[4]\}$, ε)

 insert_element(M , $\{V[2], V[3], V[4]\}$, ε)

end if

end

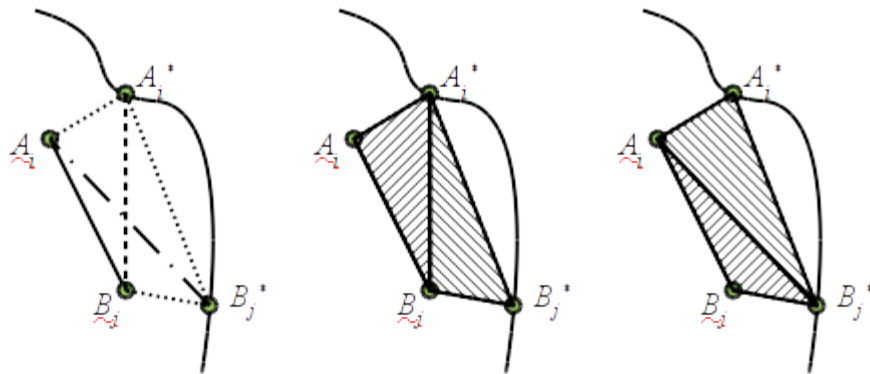


Рисунок 1.90 – Способи побудови трикутників у межевому шарі

У випадку, якщо елементами фонові дискретної моделі є тетраедри, при розбитті призми, яка утворена трикутною гранню та її проекцією, необхідно забезпечити топологічну коректність ребер та граней. Тобто, для кожної трійки вузлів, які утворюють внутрішню грань у одному елементі, повинна існувати грань з цих вузлів у іншому елементі.

Припустимо, що кодом кожного вузла є його номер у відповідній множині. Водночас будемо вважати, що у призматичних елементах порядок обходу вершин такий, як на рис. 1.91.

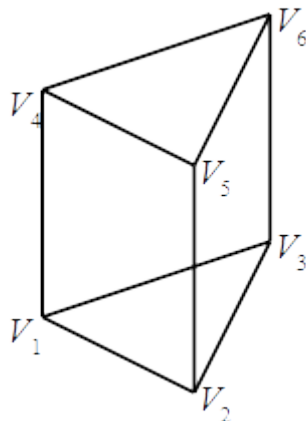


Рисунок 1.91 – Порядок обходу вершин призматичного елемента

Кожна чотирикутна грань призми може бути розбита на трикутники двома способами. У результаті існує вісім способів проведення діагоналей у чотирикутних гранях. Водночас у шести випадках можна розбити призму на три тетраедри (рис. 1.92). А у двох, що залишилися, за необхідне буде додавання нових вузлів [310].

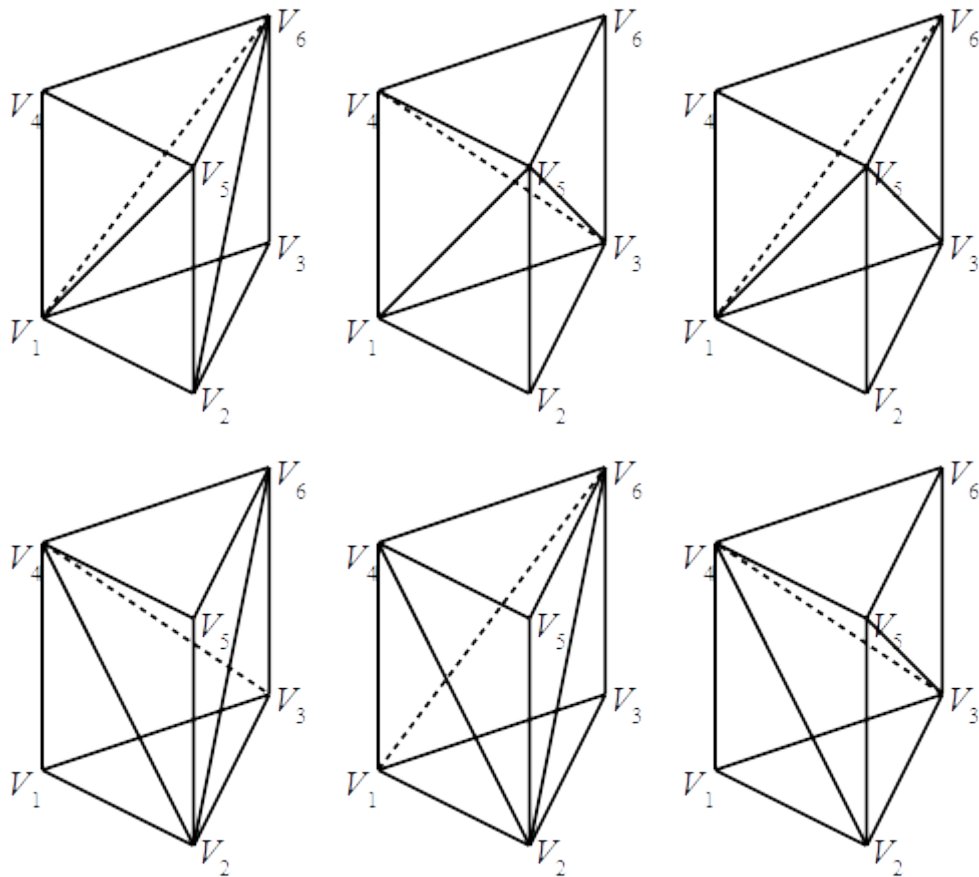


Рисунок 1.92 – Способи розбиття призми на тетраедри

Якщо прийняти вимогу, що діагоналі повинні проходити у елементі через вершину з найменшим кодом вузла, тоді у двох з трьох чотирикутних граней діагоналі будуть визначені. Щоб обрати напрям діагоналі на грані, яка залишилась, також необхідно знайти вершину з мінімальним кодом вузла та провести діагональ через неї [310]. У результаті можна отримати такий алгоритм додавання прикордонних тетраедричних елементів:

algorithm add_elements

input:

M – дискретна модель на базі тетраедрів

V – список вершин

ϵ – чисельний нуль

begin

I $\leftarrow ((1, 2, 3, 4, 5, 6), (2, 3, 1, 5, 6, 4), (3, 1, 2, 6, 4, 5), (4, 6, 5, 1, 3, 2), (5, 4, 6,$

2, 1, 3), (6, 5, 4, 3, 2, 1))

$r \leftarrow \underset{i}{\operatorname{argmin}}(\mathbf{V}[i])$

if $\min(\mathbf{V}[\mathbf{I}[r, 2]], \mathbf{V}[\mathbf{I}[r, 6]]) < \min(\mathbf{V}[\mathbf{I}[r, 3]], \mathbf{V}[\mathbf{I}[r, 5]])$ **then**

insert_element($\mathbf{M}$, { $\mathbf{V}[\mathbf{I}[r, 1]]$, $\mathbf{V}[\mathbf{I}[r, 2]]$, $\mathbf{V}[\mathbf{I}[r, 3]]$, $\mathbf{V}[\mathbf{I}[r, 6]]$ }, ε)

insert_element($\mathbf{M}$, { $\mathbf{V}[\mathbf{I}[r, 1]]$, $\mathbf{V}[\mathbf{I}[r, 2]]$, $\mathbf{V}[\mathbf{I}[r, 6]]$, $\mathbf{V}[\mathbf{I}[r, 5]]$ }, ε)

insert_element($\mathbf{M}$, { $\mathbf{V}[\mathbf{I}[r, 1]]$, $\mathbf{V}[\mathbf{I}[r, 5]]$, $\mathbf{V}[\mathbf{I}[r, 6]]$, $\mathbf{V}[\mathbf{I}[r, 4]]$ }, ε)

else

insert_element($\mathbf{M}$, { $\mathbf{V}[\mathbf{I}[r, 1]]$, $\mathbf{V}[\mathbf{I}[r, 2]]$, $\mathbf{V}[\mathbf{I}[r, 3]]$, $\mathbf{V}[\mathbf{I}[r, 5]]$ }, ε)

insert_element($\mathbf{M}$, { $\mathbf{V}[\mathbf{I}[r, 1]]$, $\mathbf{V}[\mathbf{I}[r, 5]]$, $\mathbf{V}[\mathbf{I}[r, 3]]$, $\mathbf{V}[\mathbf{I}[r, 6]]$ }, ε)

insert_element($\mathbf{M}$, { $\mathbf{V}[\mathbf{I}[r, 1]]$, $\mathbf{V}[\mathbf{I}[r, 5]]$, $\mathbf{V}[\mathbf{I}[r, 6]]$, $\mathbf{V}[\mathbf{I}[r, 4]]$ }, ε)

end if

end

Отже, розроблено метод генерації дискретних моделей форм об'єктів на базі елементів типу солід. Наприклад, для l-області (рис. 1.15), яка задана формулою (1.17), якщо використовувати як фонову рівномірну дискретну модель з 8×11 вузлів для прямокутника $[0; 30] \times [0; 40]$, то результати послідовного застосування алгоритмів background-grid-mesh і boundary_refinement (8 ітерацій згладжування дискретної моделі межі об'єкта) матимуть вигляд, наведений на рис. 1.93, а при використанні трикутників або вигляд, наведений на рис. 1.93, б, при використанні чотирикутників.

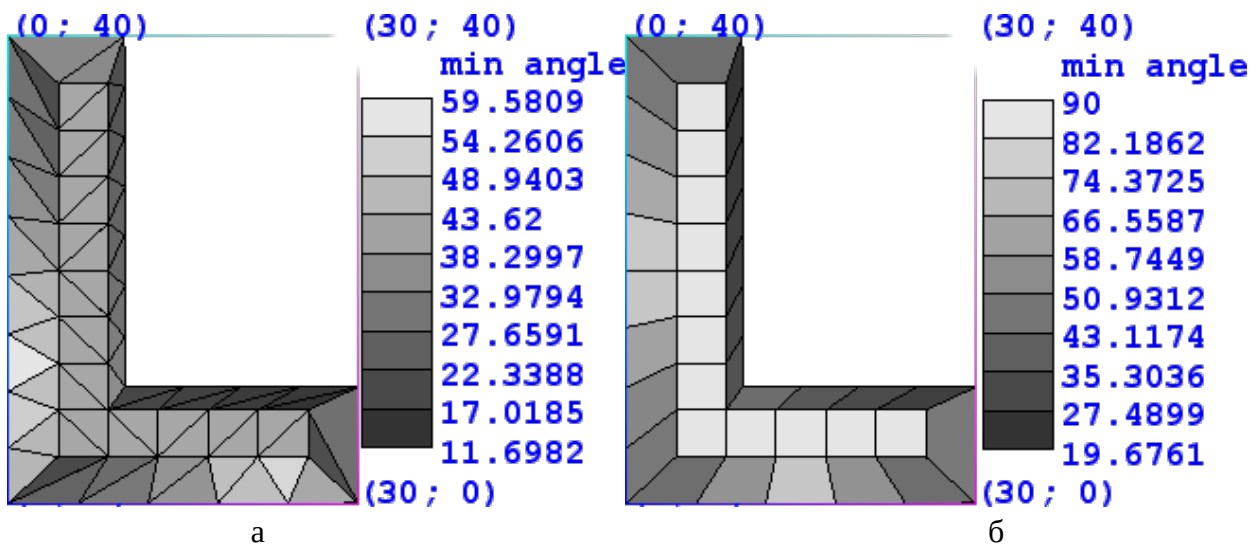


Рисунок 1.93 – Результати застосування узагальненого методу генерації дискретних моделей форм об'єктів на базі двовимірних елементів типу солід: а – дискретна модель L-області на базі трикутників; б – дискретна модель L-області на базі чотирикутників

В отриманих дискретних моделях мінімальний кут у елементі $11,7^\circ$ для трикутників і $19,8^\circ$ для чотирикутників (відповідні розподіли показані півтоном на рис. 1.93). Водночас у дискретній моделі на базі трикутників максимальний

кут у діапазоні від $60,7^\circ$ до $155,5^\circ$, а відношення мінімальної до максимальної довжин сторін елемента – від 0,34 до 0,99. У дискретній моделі на базі чотирикутників максимальний кут змінюється від 90° до $160,9^\circ$, а відношення мінімальної до максимальної довжин сторін – від 0,4 до 0,93.

Аналіз значень мінімального кута ($\alpha_{\min}$), максимального кута ($\alpha_{\max}$) і мінімального значення відношення мінімальної до максимальної довжини

сторін елемента $\left(\min \left(\frac{l_{\min}}{l_{\max}} \right) \right)$ при різній кількості елементів у фоновій

дискретної моделі показує, що можливим є отримання елементів з малими кутами як на базі трикутників, так і на базі чотирикутників (табл. 1.1).

Таблиця 1.1 – Розподіл характеристик дискретних моделей L-області

Фонова дискретна модель	8×11	9×12	10×13	11×14	11×15	12×16	13×17	14×18	14×19
Трикутні елементи									
Кількість елементів	76	88	100	144	152	168	184	242	252
$\alpha_{\min}, ^\circ$	11,7	15,9	17,7	6,9	11,9	16,0	17,5	6,9	12,3
$\alpha_{\max}, ^\circ$	155,5	144,4	135,9	166,0	154,9	144,0	135,2	166,0	154,2
$\min(l_{\min} / l_{\max})$	0,34	0,44	0,44	0,29	0,33	0,45	0,43	0,31	0,33
Чотирикутні елементи									
Кількість елементів	38	44	50	72	76	84	92	121	126
$\alpha_{\min}, ^\circ$	19,8	34,7	44,9	14,4	19,3	34,4	44,9	14,3	19,1
$\alpha_{\max}, ^\circ$	160,9	145,9	135,3	166,0	161,1	146,0	135,2	166,0	161,2
$\min(l_{\min} / l_{\max})$	0,4	0,37	0,34	0,38	0,5	0,54	0,50	0,34	0,44

При використанні чотирикутних (та шестигранних) фонових дискретних моделей можлива поява вироджених елементів (у яких є кути $\geq 180^\circ$). Наприклад, якщо використовувати як фонову рівномірну дискретну модель з 10×10 квадратних елементів для квадратної області $[-20; 20] \times [-20; 20]$, при побудові моделі пластинки, яка задана формулою (1.18), буде отримано три вироджених елементи (рис. 1.94).

У тривимірному випадку результати аналогічні двовимірним. Наприклад, для області, яка задана формулою (1.58), при використанні рівномірної дискретної моделі на базі тетраєдрів ($21 \times 21 \times 21$ вузол) для куба

$[-1;1] \times [-1;1] \times [-1;1]$ мінімальний кут в елементі змінюється від $0,62^\circ$ до $50,1^\circ$, максимальний кут – від $68,9^\circ$ до $177,2^\circ$, відношення мінімальної до максимальної довжин ребер елемента – від 0,067 до 0,81 (рис. 1.95, а). Якщо використати кубічні фонові елементи, то мінімальний кут буде у діапазоні від $7,48^\circ$ до 90° , максимальний кут – від $68,9^\circ$ до $177,2^\circ$ й відношення мінімальної до максимальної довжин ребер елемента – від 0,12 до 1 (рис. 1.95, б).

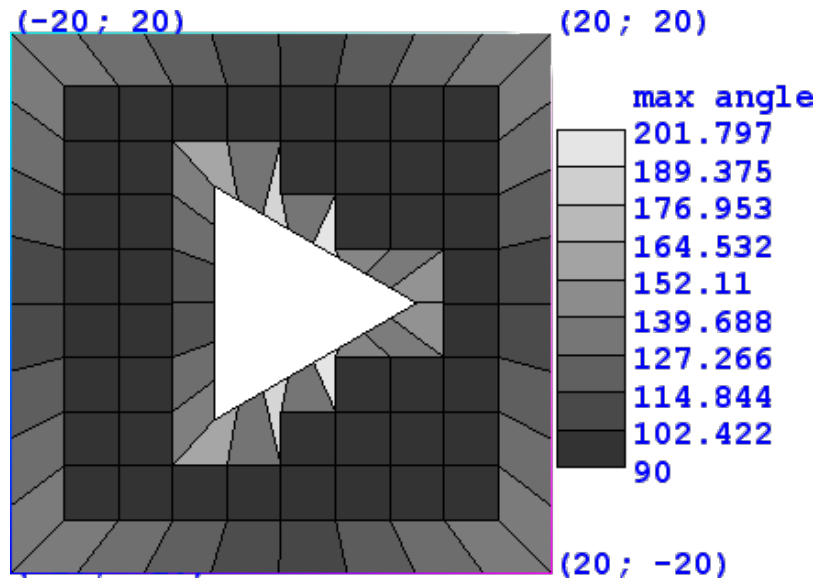
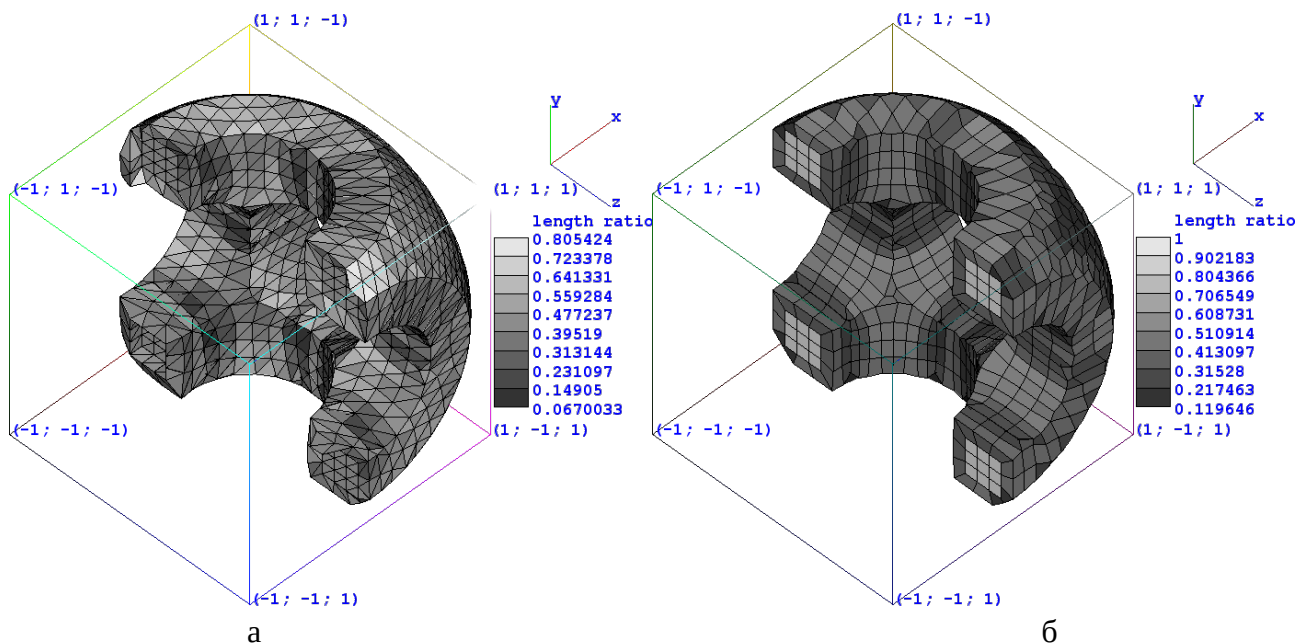


Рисунок 1.94 – Дискретна модель пластинки



б

Рисунок 1.95 – Результати застосування узагальненого методу генерації дискретних моделей форм об'єктів на базі тривимірних елементів типу солід: а – дискретна модель форми об'єкта на базі тетраедрів; б – дискретна модель форми об'єкта на базі шестигранників

У табл. 1.2 зіставляються значення мінімального кута ($\alpha_{\min}$), максимального кута ($\alpha_{\max}$) і мінімального значення відношення мінімальної до

Математичне забезпечення інженерного аналізу об'єктів аерокосмічної техніки на базі хмарних технологій 147

максимальної довжин сторін елемента $\left(\min(l_{\min}/l_{\max})\right)$ при різній кількості елементів у фонових дискретних моделях на базі тетраедрів і шестигранників.

Таблиця 1.2 – Розподіл характеристик дискретних моделей форми об'єкта, заданої формулою (1.58)

Фонові дискретні моделі	21×21	22×22	23×23	24×24	25×25	26×26	27×27	28×28	29×29
Тетраедричні елементи									
Кількість елементів	25944	30048	34704	38364	45372	51798	56328	62718	73308
$\alpha_{\min}, ^\circ$	0,6	1,2	0,5	0,5	0,1	0,3	0,7	0,8	0,5
$\alpha_{\max}, ^\circ$	177,2	177,2	178,5	178,7	179,8	179,2	177,8	177,8	177,9
$\min(l_{\min} / l_{\max})$	0,06	0,1	0,01	0,03	0,02	0,06	0,06	0,1	0,04
Шестигранні елементи									
Кількість елементів	4416	5112	5912	6536	7736	8816	9584	10688	12496
$\alpha_{\min}, ^\circ$	7,5	14,1	8,8	1,2	2,2	5,7	6,5	12,5	2,3
$\alpha_{\max}, ^\circ$	177,2	177,6	179,8	179,7	179,3	179,5	176,9	178,5	178,1
$\min(l_{\min} / l_{\max})$	0,12	0,09	0,08	0,07	0,05	0,09	0,07	0,12	0,06

Так, можна зробити висновки про те, що отримані дискретні моделі потребують подальшої обробки (наприклад, згладжування) для покращення їх чисельних характеристик. Водночас дискретні моделі на базі чотирикутних або шестигранних елементів можуть містити вироджені елементи, для усунення яких необхідно виконати додаткову обробку.

1.11.3 Згладжування Лапласа для внутрішніх вузлів

Поширеним методом покращення характеристик дискретних моделей є згладжування Лапласа для внутрішніх вузлів. Вище викладено його теоретичне обґрунтування. Алгоритмічно при згладжуванні Лапласа кожен внутрішній вузол дискретної моделі форму об'єкта переноситься у позицію середнього арифметичного сусідніх вузлів. У результаті можна записати такий алгоритм:

algorithm laplacian-smoothing**input:****M** – дискретна модель форми об'єкта*iter* – кількість ітерацій**begin****for** *i* ← 1 **to** *iter* **do****for each** *n* ∈ **A.N** **do****if** *n* ∉ δ(**M**) **then****A** ← **M**.adjacent(**M**.code(*n*))

$$n.v \leftarrow \frac{1}{|A|} \sum_{k \in A} M[k]$$

end if**end for****end for****end**

На рис. 1.96, а–г показані результати, відповідно, однієї, двох, трьох і чотирьох ітерацій згладжування Лапласа для дискретної моделі І-області на базі трикутників (рис. 1.93, а). Водночас збільшення числа ітерацій згладжування Лапласа після чотирьох ітерацій слабо впливає на основні характеристики дискретної моделі форми об'єкта (табл. 1.3).

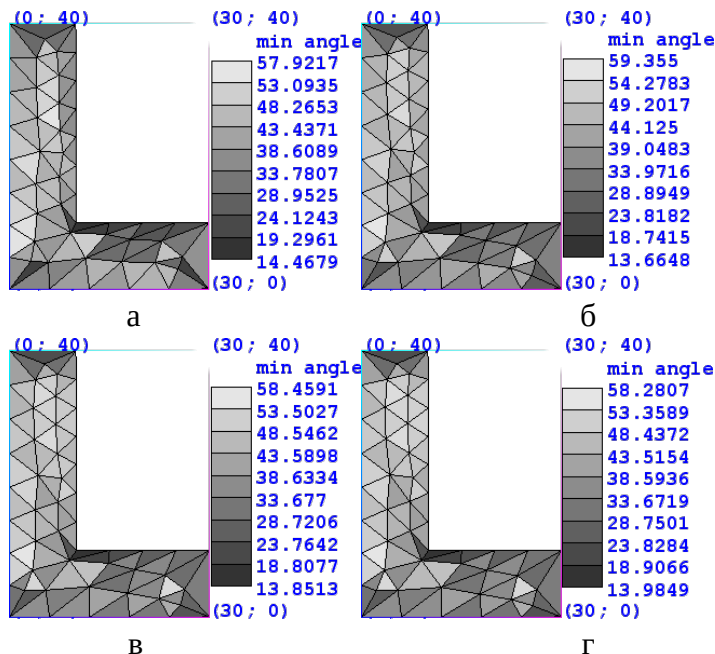


Рисунок 1.96 – Згладжування Лапласа дискретної моделі на базі трикутників:

а – 1 ітерація; б – 2 ітерації; в – 3 ітерації; г – 4 ітерації

На рис. 1.97, а–г показані результати, відповідно, однієї, двох, трьох і чотирьох ітерацій згладжування Лапласа для дискретної моделі L-області на базі чотирикутників (рис. 1.93, б). Водночас (як й у випадку з трикутниками)

Математичне забезпечення інженерного аналізу об'єктів аерокосмічної техніки на базі хмарних технологій 149

збільшення числа ітерацій згладжування Лапласа після чотирьох ітерацій слабо впливає на основні характеристики дискретної моделі форми об'єкта (табл. 1.4).

Таблиця 1.3 – Розподіл характеристик дискретної моделі L-області на базі трикутних елементів

Кількість ітерацій	0	1	2	3	4	5	6	7	8	9
$\alpha_{\min}, ^\circ$	11,7	14,5	13,7	13,9	14,0	14,1	14,2	14,2	14,2	14,2
$\alpha_{\max}, ^\circ$	155,5	129,6	134,6	136,5	137,2	137,5	137,6	137,6	137,6	137,7
$\min(l_{\min}/l_{\max})$	0,34	0,28	0,29	0,29	0,29	0,29	0,29	0,29	0,29	0,29

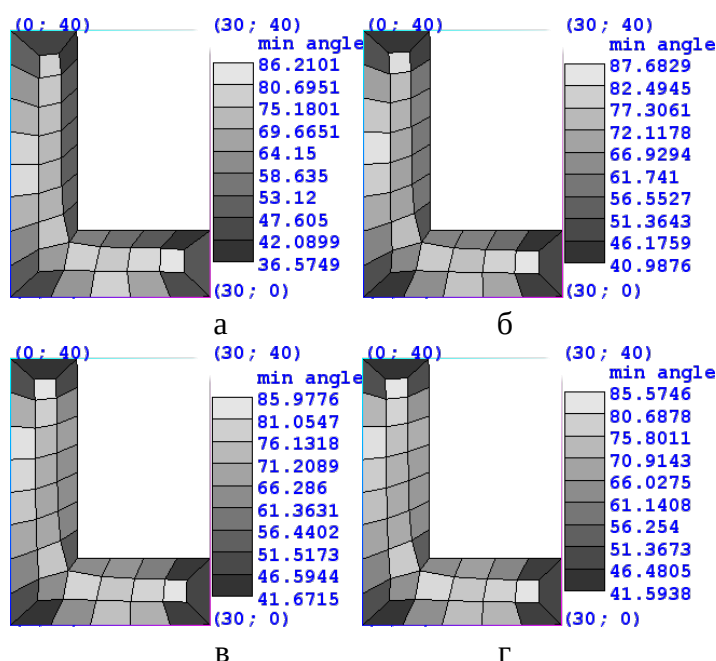


Рисунок 1.97 – Згладжування Лапласа дискретної моделі на базі чотирикутників:
а – 1 ітерація; б – 2 ітерації; в – 3 ітерації; г – 4 ітерації

Таблиця 1.4 – Розподіл характеристик дискретної моделі L-області на базі чотирикутних елементів

Кількість ітерацій	0	1	2	3	4	5	6	7	8	9
$\alpha_{\min}, ^\circ$	19,8	36,6	41,0	41,7	41,6	41,5	41,2	41	40,9	40,9
$\alpha_{\max}, ^\circ$	160,9	146,4	144,3	143,4	143,2	143,3	143,3	143,3	143,2	143,2
$\min(l_{\min}/l_{\max})$	0,4	0,3	0,32	0,33	0,33	0,33	0,33	0,33	0,33	0,33

У тривимірному випадку результати застосування згладжування Лапласа є аналогічними двовимірним. На рис. 1.98, а–г показані результати, відповідно,

однієї, двох, трьох і чотирьох ітерацій згладжування Лапласа для дискретної моделі на базі тетраедрів, яка зображена на рис. 1.95, а.

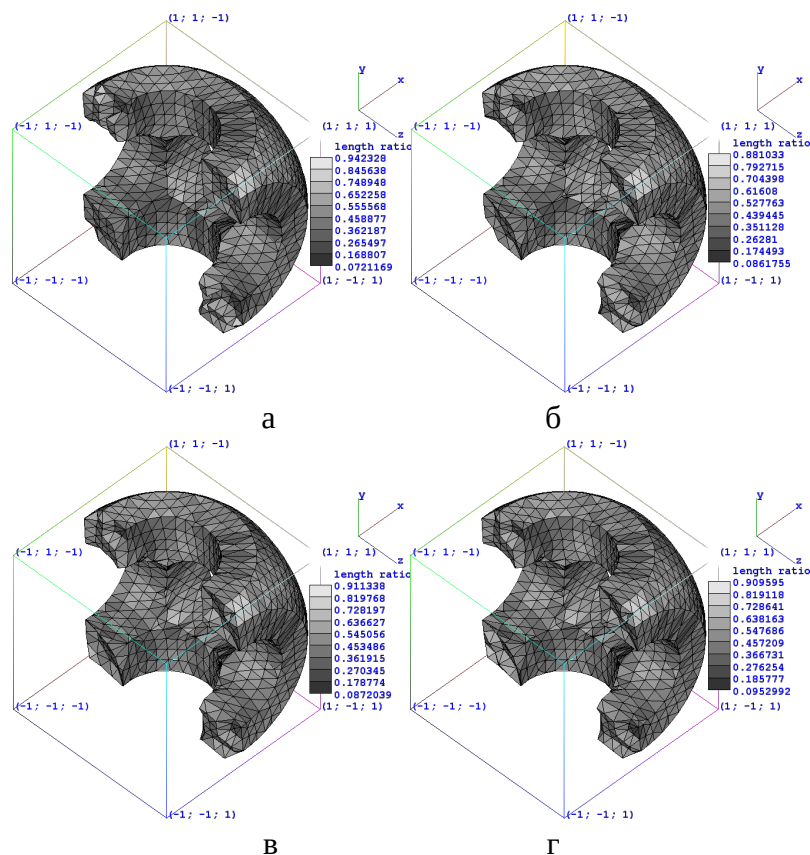


Рисунок 1.98 – Згладжування Лапласа дискретної моделі на базі тетраедрів:
а – 1 ітерація; б – 2 ітерації; в – 3 ітерації; г – 4 ітерації

При використанні більш, ніж чотирьох ітерацій згладжування Лапласа для дискретних моделей на базі тетраедрів характеристики мінімального і максимального кута, відношення довжин сторін змінюються слабо (табл. 1.5). Водночас характеристики дискретної моделі істотно залежать від відповідних значень для елементів на межі. Якщо застосувати, наприклад, чотири ітерації згладжування межових вузлів разом зі згладжуванням внутрішніх вузлів, то мінімальне значення для відношення довжин сторін буде рівним 0,23 (рис. 1.99).

Таблиця 1.5 – Розподіл значень характеристик дискретної моделі форми об'єкта на базі тетраедричних елементів

Кількість ітерацій	0	1	2	3	4	5	6	7	8	9
$\alpha_{\min}, ^\circ$	0,6	3,4	3,3	4,0	3,6	3,7	3,4	3,2	3,5	3,2
$\alpha_{\max}, ^\circ$	177,2	161,3	169,1	168,3	169,7	169,2	170,2	170,7	170,4	170,7
$\min(l_{\min}/l_{\max})$	0,06	0,07	0,09	0,09	0,11	0,11	0,11	0,11	0,11	0,12

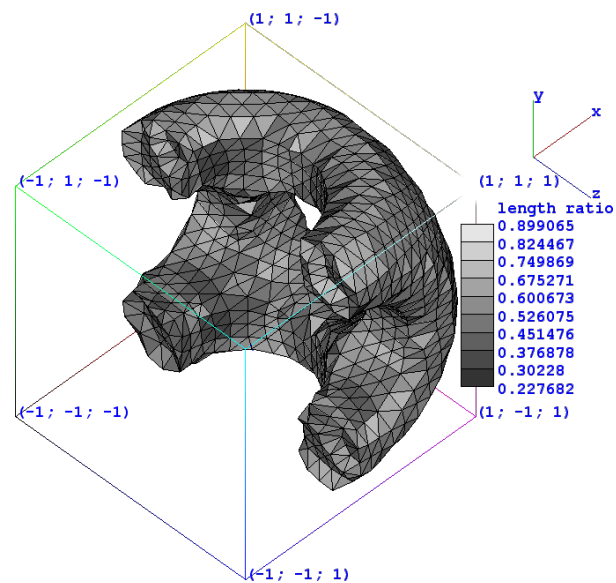
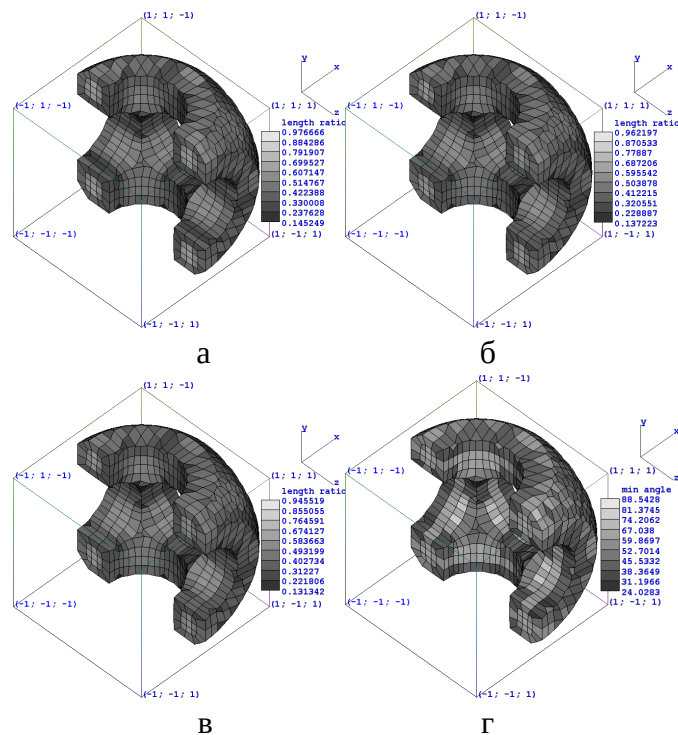


Рисунок 1.99 – Результат одночасного згладжування межових і внутрішніх вузлів

На рис. 1.100, а–г показані результати, відповідно, однієї, двох, трьох і чотирьох ітерацій згладжування Лапласа для дискретної моделі на базі шестигранників, зображеної на рис. 1.95, б. Як і в разі тетраєдрів, при використанні більш чотирьох ітерацій згладжування Лапласа для дискретних моделей на базі шестигранників характеристики мінімального і максимального кута, відношення довжин сторін змінюються слабо (табл. 1.6): починаючи з п'ятої ітерації мінімальний кут знаходиться в інтервалі $23\text{--}24^\circ$, а мінімальне співвідношення довжин сторін в елементі в межах 0,12–0,13.

Рисунок 1.100 – Згладжування Лапласа дискретної моделі на базі шестигранників:
а – 1 ітерація; б – 2 ітерації; в – 3 ітерації; г – 4 ітерації

Таблиця 1.6 – Розподіл значень характеристик дискретної моделі форми об'єкта на базі шестигранних елементів

Кількість ітерацій	0	1	2	3	4	5	6	7	8	9
$\alpha_{\min}, ^\circ$	7,5	15,8	24,0	24,2	24,0	23,8	23,7	23,5	23,4	23,3
$\alpha_{\max}, ^\circ$	177,2	178,4	178,4	178,4	179,5	179,2	179,1	179,0	179,0	178,9
$\min(l_{\min}/l_{\max})$	0,12	0,15	0,15	0,15	0,13	0,13	0,12	0,12	0,12	0,12

Як і для тетраедрів, характеристики дискретної моделі на базі шестигранників залежать від положень вузлів на межі. Якщо застосувати чотири ітерації згладжування і внутрішніх, і межових вузлів, тоді мінімальне значення для відношення довжин сторін буде приблизно рівним 0,21 (рис. 1.101).

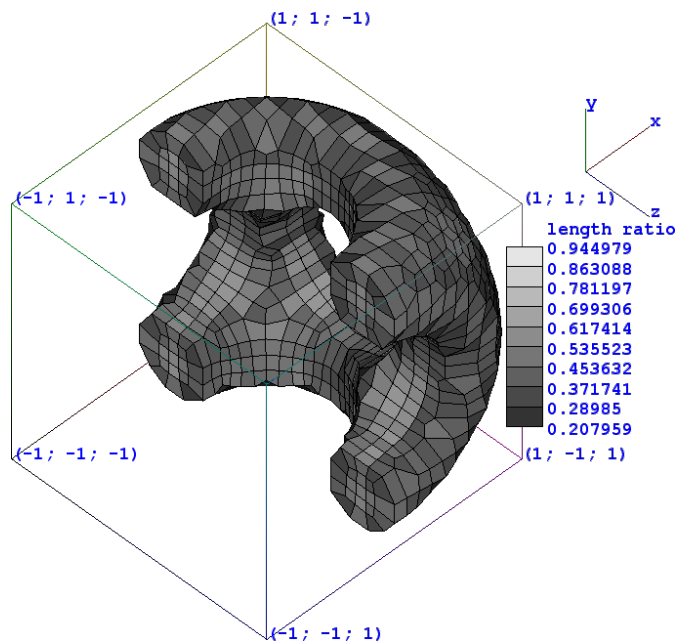


Рисунок 1.101 – Результат одночасного згладжування межових і внутрішніх вузлів дискретної моделі форми об'єкта на базі шестигранників

1.11.4 Метод згладжування координат внутрішніх вузлів дискретних моделей форм об'єктів

Практичне застосування згладжування Лапласа зумовлює поступове вирівнювання довжин внутрішніх ребер елементів. Проте, переміщення вузлів в координати геометричних центрів опуклих фігур, які утворюються сусідніми вузлами, може не тільки не покращувати характеристики дискретної моделі

форми об'єкта, але й спричинити появу вироджених (у разі використання чотирикутних або шестигранних елементів). У такому випадку можна використовувати згладжування, засноване на мінімізації деякої цільової функції, визначеної на вузлах дискретної моделі форми об'єкта.

Розглянемо плоский чотирикутний елемент. Аплікати векторних добутків пар векторів, направлених з кожної вершини уздовж суміжних у ній ребер (рис. 1.102), дорівнюють подвоєним площам трикутників, визначених вершиною, суміжними ребрами і діагоналлю. Причому знак аплікати буде додатним, якщо кут між векторами набуває значення більше 0 і менше π ; від'ємним, якщо кут між векторами від π до 2π . Апліката дорівнюватиме нулю, якщо вектори лежать на одній прямій. Отже, у неопуклого чотирикутника хоча б в одному вузлі такий добуток буде меншим або дорівнювати нулю.

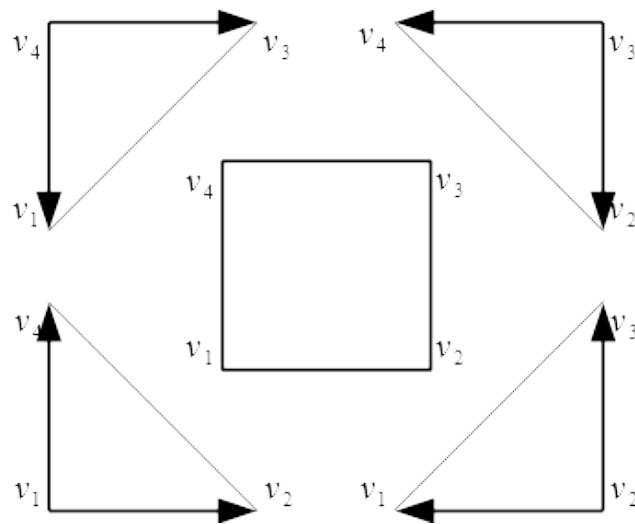


Рисунок 1.102 – Подання чотирикутника чотирма трикутниками

Для кожного елемента дискретної моделі $\mathbf{M}$, код якого k_e ($n_e = |\mathbf{M}(k_e)|$ – кількість вершин у елементі), можна визначити функціонал вигляду

$$L(\mathbf{M}, k_e) = \sum_{j=1}^{n_e} e^{-t \alpha(\mathbf{M}(k_e, j-1), \mathbf{M}(k_e, j), \mathbf{M}(k_e, j+1))}, \quad (1.63)$$

де α – площа трикутника зі знаком, яка визначена такою формулою:

$$\begin{aligned} \alpha(\mathbf{M}(k_e, j-1), \mathbf{M}(k_e, j), \mathbf{M}(k_e, j+1)) &= \\ &= (\mathbf{M}(k_e, j-1) - \mathbf{M}(k_e, j)) \times (\mathbf{M}(k_e, j+1) - \mathbf{M}(k_e, j)). \end{aligned} \quad (1.64)$$

Доданки у формулі (1.63) будуть приймати значення від 0 до 1, якщо відповідний кут більший 0 і менший π , дорівнювати 1, якщо кут, рівний 0 або π , і є більшими 1, якщо кут від π до 2π . У результаті функціонал, визначений на

множині елементів $\mathbf{E}$ дискретної моделі $\mathbf{M}$, мінімізація якого дозволить усунути елементи з кутом, більшим або рівним π , матиме такий вигляд ($\mathbf{M}.\text{code}(e)$ – код елемента e):

$$F(\mathbf{M}) = \sum_{e \in \mathbf{M}.\mathbf{E}} L(\mathbf{M}, \mathbf{M}.\text{code}(e)). \quad (1.65)$$

Значення параметру t обирається з врахуванням розмірів об'єкту та кількості елементів у дискретній моделі. Можна вважати, що

$$t = \frac{1}{\bar{\alpha}},$$

де $\bar{\alpha}$ – середнє значення функції (1.64) для елементів дискретної моделі.

Мінімізація функціоналу (1.65) для всіх вузлів сітки одночасно є вельми екстенсивною процедурою, тому що кількість незалежних змінних буде в два рази більше кількості вузлів у сітці. Зменшити розмірність розв'язуваної задачі можна за допомогою послідовних локальних рішень для кожного вузла.

Для кожного вузла з кодом k_i можна визначити функціонал вигляду

$$W(\mathbf{M}, k_i) = \sum_{k_e \in \mathbf{I}_k} L(\mathbf{M}, k_e), \quad (1.66)$$

де $\mathbf{I}_k = \mathbf{M}.\text{incident}(k_i)$ – множина елементів, інцидентних у вузлі з кодом k_i .

Функціонал (1.66) містить дві незалежні змінні (координати вузла), що суттєво спрощує пошук мінімуму. Послідовну мінімізацію функціоналів (1.66) можна розглядати як ітераційне наближення до мінімуму функціоналу (1.65) (наприклад, за допомогою градієнтного спуску). За аналогією зі згладжуванням Лапласа для досягнення найкращого результату дану процедуру необхідно повторювати кілька разів. У результаті отримано такий алгоритм:

algorithm local-functional-smoothing

input:

$\mathbf{M}$ – дискретна модель форми об'єкта

W – цільова функція

$iter$ – кількість ітерацій

begin

for $i \leftarrow 1$ **to** $iter$ **do**

for each $n \in \mathbf{M}.\mathbf{N}$ **do**

if $n \notin \delta(\mathbf{M})$ **then**

$n.v \leftarrow \underset{n.v}{\operatorname{argmin}} W(\mathbf{M}, \mathbf{M}.\text{code}(n))$

```

end if
end for
end for
end

```

Розглянемо роботу описаного вище алгоритму на прикладі дискретної моделі квадратної пластинки з трикутним вирізом, форма якої задана формулою (1.18). Вихідна модель (рис. 1.103, а), отримана за допомогою алгоритму background-grid-mesh при використанні рівномірної фонові сітки з 15×15 квадратних елементів, містить вироджені елементи біля вершин вирізу. У результаті згладжування Лапласа (4 ітерації, рис. 1.103, б) загальне розташування вузлів стало краще, але залишилися три неопуклих елемента (близько вершин вирізу). Після однієї ітерації запропонованим алгоритмом (рис. 1.103, в) модель вже не містить неопуклих елементів, але близько отвору є елемент з кутом, близьким до 180° . На рис. 1.103, г, д, є і ж, відповідно, показані результати застосування 2, 4, 8 і 16 ітерацій по вузлам з локальною мінімізацією. Можна побачити, що вже на 8 ітераціях ребро біля правого кута вирізу практично збігається з його бісектрисою.

Для вихідної дискретної моделі функціонал (1.65) дорівнює 58,217. У результаті мінімізації (1.65) за допомогою методу градієнтного спуску в глобальній постановці (одночасно для всіх внутрішніх вузлів) отримана дискретна модель, для якої функціонал (1.65) дорівнює 1,451 (рис. 1.103, з). Мінімальний кут в елементах дискретної моделі буде в межах від $27,3^\circ$ до $89,6^\circ$, максимальний кут – від $90,7^\circ$ до $151,6^\circ$, а відношення довжин сторін – від 0,33 до 0,98.

Необхідно відзначити, що зовнішній кут біля вершин трикутного отвору ділиться на два одним ребром, отже, мінімально можливою оцінкою для характеристики максимального кута буде 150° . У табл. 1.7 порівнюються значення функціоналу (1.65), а також характеристики мінімального і максимального кутів, відношення довжин сторін у залежності від кількості ітерацій згладжування внутрішніх вузлів дискретної моделі.

Таблиця 1.7 – Розподіл характеристик у дискретній моделі пластинки

Кількість ітерацій	1	2	3	4	5	6	7	8	9	10
$F(\mathbf{M})$	5,423	2,796	2,233	1,965	1,809	1,710	1,643	1,598	1,565	1,542
$\alpha_{\min}, ^\circ$	33,7	34,1	33,0	32,2	31,5	30,9	30,4	29,9	29,6	29,3
$\alpha_{\max}, ^\circ$	177,9	158,9	156,6	155,5	154,7	154,2	153,8	153,5	153,2	153,0
$\min(l_{\min}/l_{\max})$	0,28	0,37	0,41	0,40	0,38	0,37	0,37	0,36	0,36	0,35

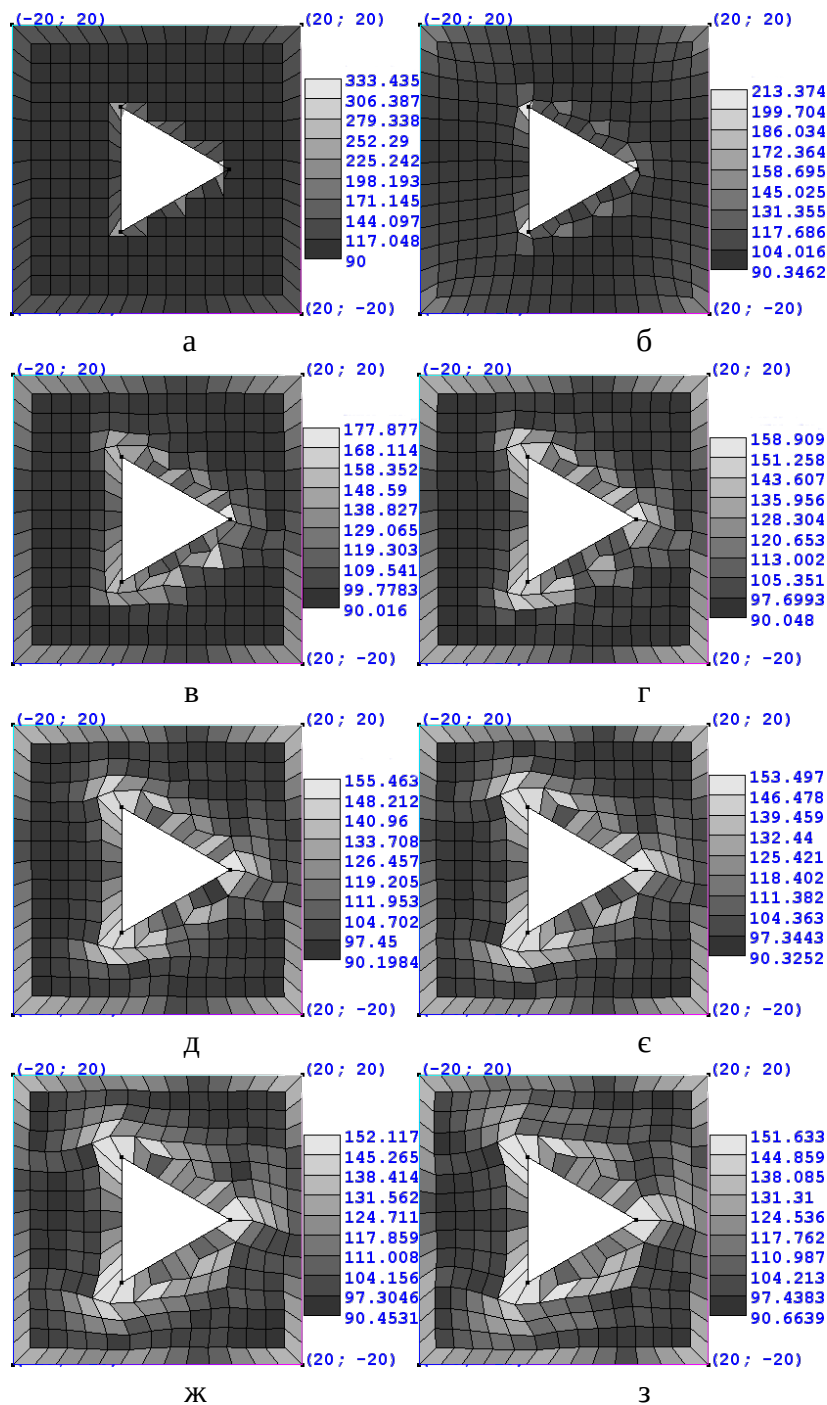


Рисунок 1.103 – Згладжування внутрішніх вузлів дискретної моделі пластинки на базі чотирикутних елементів: а – вихідна дискретна модель пластинки; б – 4 ітерації згладжування Лапласа; в – 1 ітерація згладжування на основі локальної мінімізації функціоналу; г – 2 ітерації згладжування на основі локальної мінімізації функціоналу; д – 4 ітерації згладжування на основі локальної мінімізації функціоналу; є – 8 ітерацій згладжування на основі локальної мінімізації функціоналу; ж – 16 ітерацій згладжування на основі локальної мінімізації функціоналу; з – результат глобальної мінімізації функціоналу (1.65)

При використанні фонові дискретної моделі на базі 50×50 квадратних елементів для моделі фрикційного диска трансмісії, яка задана формулою (1.20), результат буде містити вироджені чотирикутники (рис. 1.104, а). Після чотирьох

ітерацій згладжування все ще залишаються з максимальним кутом більшим 180° (рис.1.104, б). У результаті шістнадцяти ітерацій згладжування (рис. 1.104, в) вироджених елементів не буде (максимальний кут елемента знаходиться у діапазоні від $90,16^\circ$ до $167,38^\circ$). Водночас у результаті мінімізації (1.65) одночасно для всіх внутрішніх вузлів результати будуть близькими до 16 ітерацій локальної мінімізації (максимальний кут елемента знаходиться у діапазоні від $90,20^\circ$ до $167,25^\circ$).

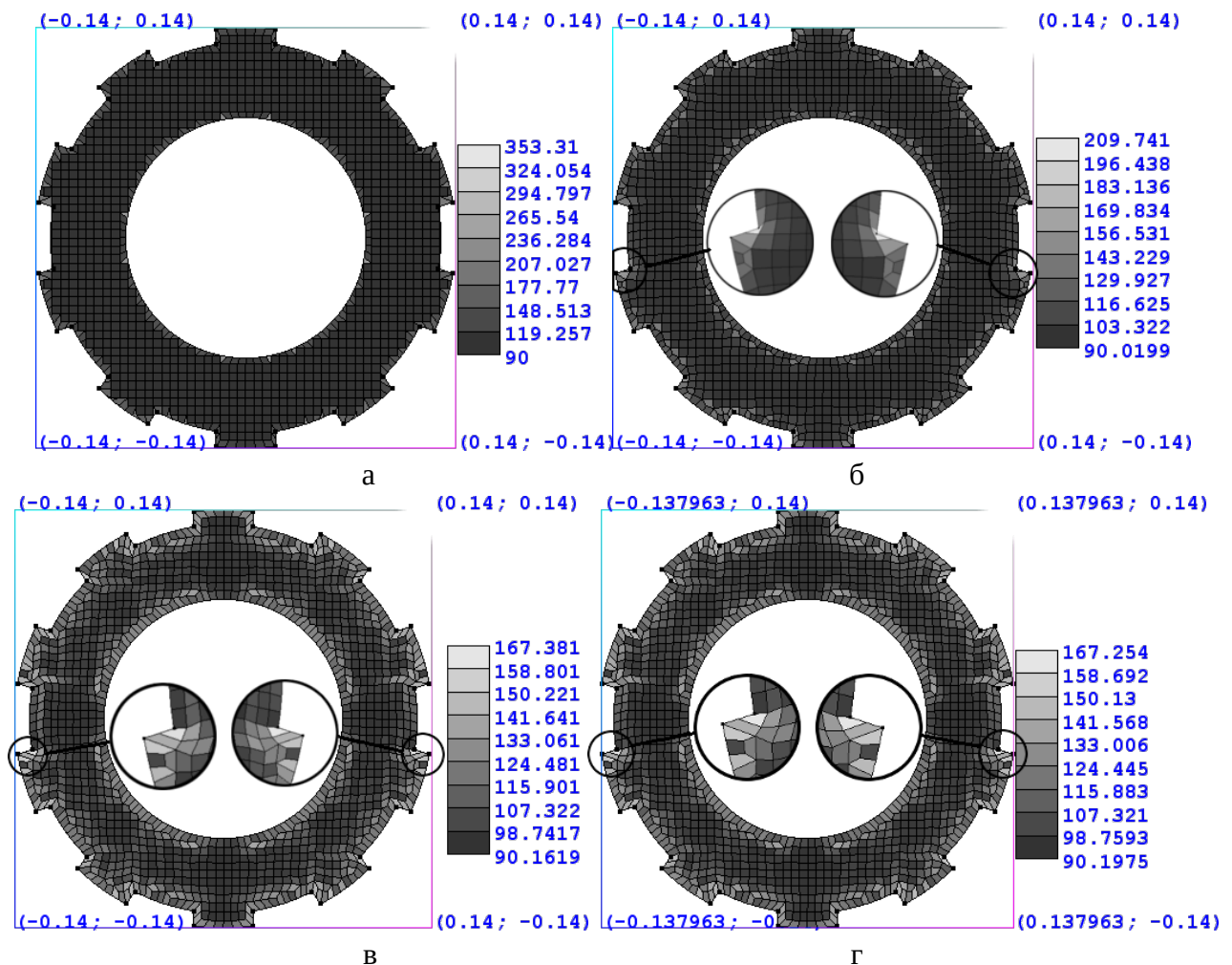


Рисунок 1.104 – Згладжування дискретної моделі форми фрикційного диска: а – вихідна дискретна модель; б – 4 ітерації згладжування Лапласа; в – 16 ітерацій згладжування на основі локальної мінімізації функціоналу; г – результат глобальної мінімізації функціоналу (1.65)

Згладжування на основі локальної мінімізації можна застосовувати у тривимірному випадку. Для цього у формулах (1.63–1.64) замість площі необхідно використовувати вирази для об'єму зі знаком. Для кожної вершини об'єм з урахуванням знака визначається як змішаний добуток трьох векторів, визначених суміжними ребрами. При використанні тетраедричних елементів таких добутоків буде чотири, а при шестигранних – вісім.

У тривимірному випадку для кожного елемента дискретної моделі $\mathbf{M}$, код якого k_e (n_e – кількість вершин у елементі), функція (1.63) матиме такий вигляд:

$$L(\mathbf{M}, k_e) = \sum_{j=1}^{n_e} e^{-t\alpha(\mathbf{M}, k_e, j)},$$

де функція α для тетраедрів визначається виразом (за умови порядку обходу вершин, зображеного на рис. 1.57, б)

$$\alpha(\mathbf{M}, k_e, j) = \begin{cases} [(\mathbf{M}(k_e, 2) - \mathbf{M}(k_e, 1)) \times (\mathbf{M}(k_e, 3) - \mathbf{M}(k_e, 1))] \cdot ((\mathbf{M}(k_e, 4) - \mathbf{M}(k_e, 1))), & j=1, \\ [(\mathbf{M}(k_e, 3) - \mathbf{M}(k_e, 2)) \times (\mathbf{M}(k_e, 4) - \mathbf{M}(k_e, 2))] \cdot ((\mathbf{M}(k_e, 1) - \mathbf{M}(k_e, 2))), & j=2, \\ [(\mathbf{M}(k_e, 2) - \mathbf{M}(k_e, 3)) \times (\mathbf{M}(k_e, 4) - \mathbf{M}(k_e, 3))] \cdot ((\mathbf{M}(k_e, 1) - \mathbf{M}(k_e, 3))), & j=3, \\ [(\mathbf{M}(k_e, 1) - \mathbf{M}(k_e, 4)) \times (\mathbf{M}(k_e, 3) - \mathbf{M}(k_e, 4))] \cdot ((\mathbf{M}(k_e, 2) - \mathbf{M}(k_e, 4))), & j=4. \end{cases}$$

Для шестигранників, якщо порядок обходу вершин відповідає зображеному на рис. 1.57, г, функція α отримає такий вигляд:

$$\alpha(\mathbf{M}, k_e, j) = \begin{cases} [(\mathbf{M}(k_e, 2) - \mathbf{M}(k_e, 1)) \times (\mathbf{M}(k_e, 4) - \mathbf{M}(k_e, 1))] \cdot ((\mathbf{M}(k_e, 5) - \mathbf{M}(k_e, 1))), & j=1, \\ [(\mathbf{M}(k_e, 6) - \mathbf{M}(k_e, 2)) \times (\mathbf{M}(k_e, 3) - \mathbf{M}(k_e, 2))] \cdot ((\mathbf{M}(k_e, 1) - \mathbf{M}(k_e, 2))), & j=2, \\ [(\mathbf{M}(k_e, 7) - \mathbf{M}(k_e, 3)) \times (\mathbf{M}(k_e, 4) - \mathbf{M}(k_e, 3))] \cdot ((\mathbf{M}(k_e, 2) - \mathbf{M}(k_e, 3))), & j=3, \\ [(\mathbf{M}(k_e, 3) - \mathbf{M}(k_e, 4)) \times (\mathbf{M}(k_e, 8) - \mathbf{M}(k_e, 4))] \cdot ((\mathbf{M}(k_e, 1) - \mathbf{M}(k_e, 4))), & j=4, \\ [(\mathbf{M}(k_e, 6) - \mathbf{M}(k_e, 5)) \times (\mathbf{M}(k_e, 1) - \mathbf{M}(k_e, 5))] \cdot ((\mathbf{M}(k_e, 8) - \mathbf{M}(k_e, 5))), & j=5, \\ [(\mathbf{M}(k_e, 7) - \mathbf{M}(k_e, 6)) \times (\mathbf{M}(k_e, 2) - \mathbf{M}(k_e, 6))] \cdot ((\mathbf{M}(k_e, 5) - \mathbf{M}(k_e, 6))), & j=6, \\ [(\mathbf{M}(k_e, 6) - \mathbf{M}(k_e, 7)) \times (\mathbf{M}(k_e, 8) - \mathbf{M}(k_e, 7))] \cdot ((\mathbf{M}(k_e, 3) - \mathbf{M}(k_e, 7))), & j=7, \\ [(\mathbf{M}(k_e, 7) - \mathbf{M}(k_e, 8)) \times (\mathbf{M}(k_e, 5) - \mathbf{M}(k_e, 8))] \cdot ((\mathbf{M}(k_e, 4) - \mathbf{M}(k_e, 8))), & j=8. \end{cases}$$

Наприклад, якщо як вихідну використовувати дискретну модель на базі тетраедрів (рис. 1.95, а) для форми об'єкта, заданого формулою (1.58), то у результаті чотирьох ітерацій згладжування внутрішніх вузлів на основі локальної мінімізації функціоналу максимальний кут у елементах буде знаходитись у діапазоні від $66,7^\circ$ до $151,2^\circ$ (рис. 1.105, а), в результаті восьми ітерацій – від $65,0^\circ$ до $149,6^\circ$ (рис. 1.105, б), а в результаті шістнадцяти ітерацій – від $63,1^\circ$ до $149,6^\circ$ (рис. 1.105, в). Значення основних характеристик дискретної моделі у залежності від кількості ітерацій такого згладжування показані у табл. 1.8.

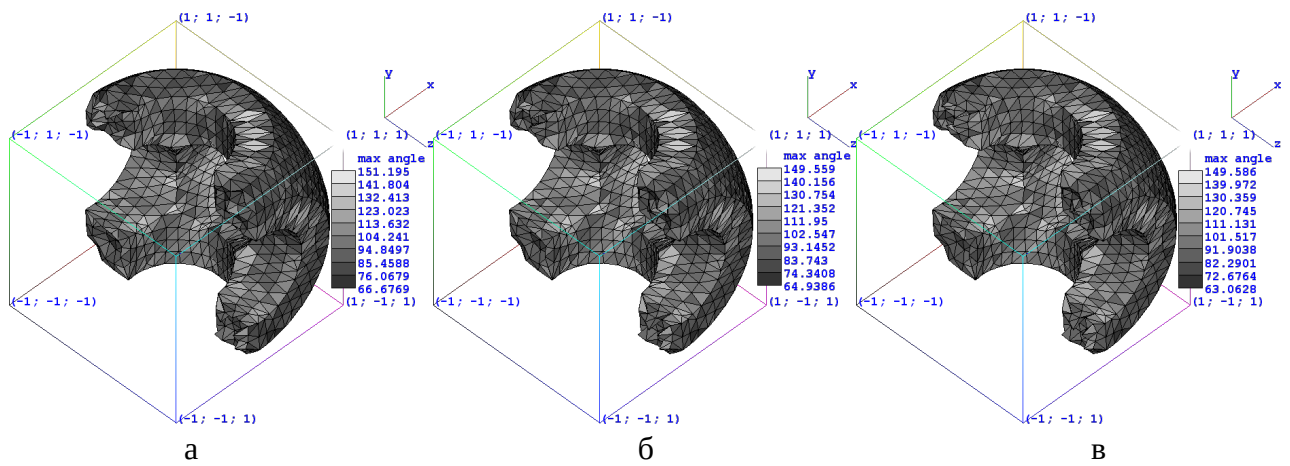


Рисунок 1.105 – Згладжування дискретної моделі на базі тетраедрів: а – 4 ітерації згладжування на основі локальної мінімізації функціоналу; б – 8 ітерацій згладжування на основі локальної мінімізації функціоналу; в – 16 ітерацій згладжування на основі локальної мінімізації функціоналу

Таблиця 1.8 – Розподіл характеристик дискретної моделі на базі тетраедрів при згладжуванні з використанням локальної мінімізації функціоналу

Кількість ітерацій	0	1	2	3	4	5	6	7	8	9
$\alpha_{\min}, ^\circ$	0,6	4,7	4,3	4,2	4,1	4,1	4,1	4,1	4,0	4,0
$\alpha_{\max}, ^\circ$	177,2	157,7	152,4	151,4	151,2	150,7	150,2	149,8	149,6	149,6
$\min(l_{\min}/l_{\max})$	0,06	0,085	0,078	0,076	0,075	0,075	0,074	0,074	0,073	0,073

Якщо як вихідну використовувати дискретну модель на базі шестигранників (рис. 1.95, б) для форми об'єкта, заданого формулою (1.58), то у результаті чотирьох ітерацій згладжування внутрішніх вузлів на основі локальної мінімізації функціоналу максимальний кут в елементах буде знаходитися в діапазоні від $92,7^\circ$ до $162,2^\circ$ (рис. 1.106, а), у результаті восьми ітерацій – від $93,2^\circ$ до $162,2^\circ$ (рис. 1.106, б), а у результаті шістнадцяти ітерацій – від $93,7^\circ$ до $162,2^\circ$ (рис. 1.106, в). Зазначимо, що значення $162,2^\circ$ відповідає найбільшому кутку межевої грані. Значення основних характеристик дискретної моделі у залежності від кількості ітерацій такого згладжування показані у табл. 1.9.

Таблиця 1.9 – Розподіл характеристик дискретної моделі на базі шестигранників при згладжуванні з використанням локальної мінімізації функціоналу

Кількість ітерацій	0	1	2	3	4	5	6	7	8	9
$\alpha_{\min}, ^\circ$	7,5	15,4	24,1	27,2	28,3	28,3	28,3	28,3	28,3	28,3

Кількість ітерацій	0	1	2	3	4	5	6	7	8	9
$\alpha_{\max}, ^\circ$	177,2	170,6	162,2	162,2	162,2	162,2	162,2	162,2	162,2	162,2
$\min(l_{\min}/l_{\max})$	0,12	0,13	0,13	0,13	0,13	0,13	0,13	0,13	0,13	0,13

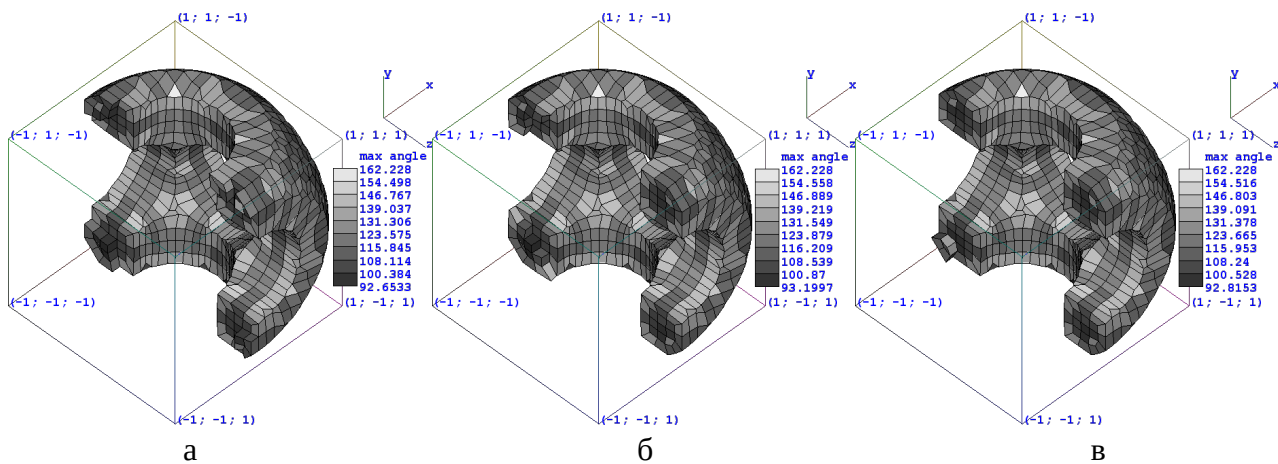


Рисунок 1.106 – Згладжування дискретної моделі на базі шестигранників: а – 4 ітерації згладжування на основі локальної мінімізації функціоналу; б – 8 ітерацій згладжування на основі локальної мінімізації функціоналу; в – 16 ітерацій згладжування на основі локальної мінімізації функціоналу

Для порівняльної оцінки розглянемо час, необхідний на згладжування Лапласа, локальну і глобальну мінімізації функціоналу (1.65), у залежності від числа вузлів у дискретній моделі пластинки з трикутним отвором (рис. 1.103). Для обчислювального експерименту використовувався комп'ютер з процесором Intel Core i5-3330 3.0GHz (4 ядра) і 8 гігабайтами оперативної пам'яті (результати показані у табл. 1.10). Під час обчислювального експерименту процедури згладжування Лапласа і локальної мінімізації з використанням запропонованого підходу повторювалися у чотирьох ітераціях. Можна побачити, що найменший час необхідний для виконання згладжування Лапласа. Локальна мінімізація у середньому на три порядки довше згладжування Лапласа і на три порядки швидше пошуку мінімуму у глобальній постановці.

Таблиця 1.10 – Залежність часу згладжування від кількості вузлів у дискретній моделі пластинки з трикутним отвором

Метод згладжування	Кількість вузлів						
	828	1138	1496	1912	2367	2874	3427
	Час, с						
Згладжування Лапласа	0,002	0,003	0,004	0,005	0,006	0,007	0,009

Метод згладжування	Кількість вузлів						
	828	1138	1496	1912	2367	2874	3427
	Час, с						
Локальна мінімізація	0,382	0,469	0,880	1,108	1,420	1,710	2,029
Глобальна мінімізація	63,620	102,300	271,000	428,316	662,700	947,434	1338,041

Отже, розроблений метод згладжування внутрішніх вузлів дискретних моделей на основі локальної мінімізації функціоналу дозволяє не тільки покращити їх характеристики, а й усунути вироджені елементи (при використанні чотирикутників або шестигранників). Як видно з рис. 1.103–1.106, а також з табл. 1.7–1.9, для досягнення прийнятних результатів достатньо 8-10 ітерацій. Водночас дослідження витрат часу, дозволяє зробити висновок, що локальний алгоритм мінімізації функціоналу в середньому на три порядки довший згладжування Лапласа і на три порядки швидший пошуку його мінімуму за допомогою глобального алгоритму.

1.11.5 Особливості використання методу для побудови дискретних моделей поверхонь об'єктів, поданих гібридно

При моделюванні оболонкових конструкцій у таких галузях як, наприклад, ракетобудування, зручним у використанні може бути подання поверхні у вигляді (1.53). У такому випадку вихідними даними будуть:

– $\text{surface}(u, v)$ – параметрична поверхня, визначена на прямокутнику $P = \{(u, v) | a \leq u \leq b, c \leq v \leq d\}$;

– $F(x, y, z)$ – неявна функція, яка більша нуля у внутрішніх точках поверхні об'єкта, що моделюється.

Структурно метод побудови тріангуляції такої поверхні можна подати трьома послідовними етапами:

I – побудова дискретної моделі меж поверхні;

II – побудова початкової тріангуляції;

III – оптимізація тріангуляції.

Визначимо операцію обчислення відстані між двома точками як функцію

$$d(p, q) = \|\text{surface}(p) - \text{surface}(q)\|,$$

де p і q – координати точок у параметричному просторі ($p, q \in \mathbf{P}$).

Також визначимо функцію

$$W(u, v) = F(R(u, v)) \wedge \left[\frac{(b-a)^2}{4} - \left(u - \frac{a+b}{2} \right)^2 \right] \wedge \left[\frac{(d-c)^2}{4} - \left(v - \frac{d+c}{2} \right)^2 \right],$$

яка більше або дорівнює нулю тільки у внутрішніх точках для функції F прямокутника $\mathbf{P}$.

Для подальшої побудови дискретної моделі використовуватимемо цю функцію W . Алгоритм метода запишемо у вигляді трьох кроків.

I. У параметричному просторі побудуємо дискретну модель меж об'єкта з використанням алгоритму `background-grid-mesh`. Для цього прямокутник $\mathbf{P}$ розіб'ємо на рівномірні фонові комірки (у загальному випадку можливо використовувати не рівномірне розбиття), а як функцію, яка описує форму поверхні, використаємо W .

II. Побудуємо початкову триангуляцію для моделі меж об'єкта, отриманої на попередньому кроці, за допомогою алгоритму `delaunay` з перевіркою умови Делоне у параметричному просторі. Природно, що двовимірна триангуляція Делоне, побудована у параметричному просторі, не обов'язково буде зберігати відповідність критерію Делоне, після відображення сітки на поверхню.

III. Приведемо дискретну модель у відповідність критерію Делоне за допомогою алгоритму `refine_delaunay`, у якому для перевірки потрапляння в описане коло використовується алгоритм `circum-circle` для тривимірних трикутних граней (пошук центру і радіусу здійснюється у місцевій системі координат). Водночас координати центру описаного кола у параметричному просторі визначаються формулою

$$c = \operatorname{argmin}_{p \in \mathbf{P}} d(\operatorname{surface}(p), C),$$

де C – центр описаного екваторіального кола у тривимірних координатах.

Наприклад, для поверхні Безьє з круглим отвором, яка задана формулою (1.54), початкова триангуляція у параметричному просторі та результат її покращення наведені на рис. 1.107. Водночас для моделі конічної оболонки, яка задана конічною параметричною поверхнею і функцією (1.55), що описує технологічні отвори, можна отримати дискретну модель у вигляді, наведеному рис. 1.108. Можна побачити, що в елементах моделі відсутні кути, менші 22° .

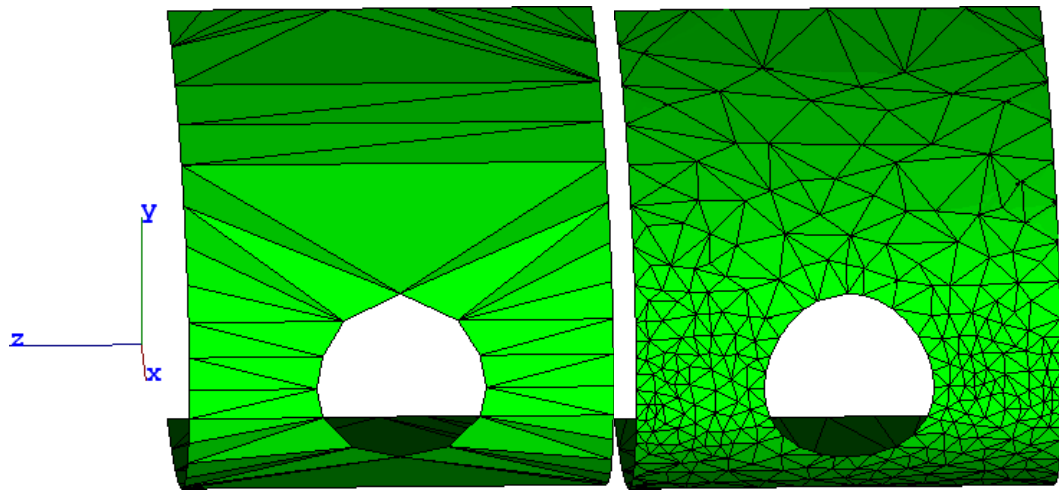


Рисунок 1.107 – Тріангуляція поверхні Безьє з круглим отвором

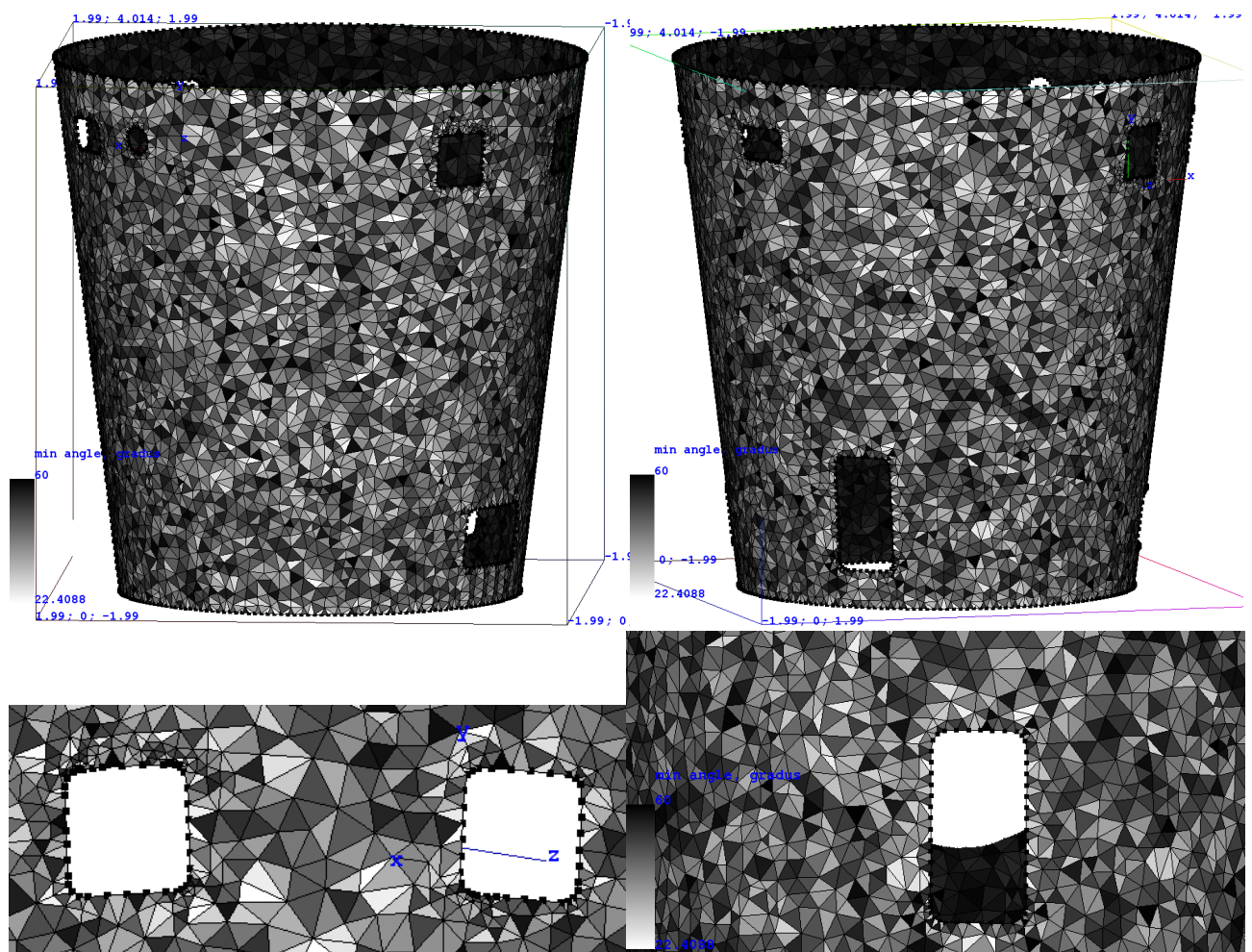


Рисунок 1.108 – Конічна оболонка: розподіл значень мінімальних кутів у трикутниках

РОЗДІЛ 2. ПРОЦЕСОР

2.1. Дослідження впливу форми скінченного елемента на точність розв'язку

В основу методу скінченних елементів [92–94] покладено ідею апроксимації шуканої неперервної величини (наприклад, температури, переміщення тощо) за допомогою кусково-неперервних функцій, які визначені на елементарних геометричних областях – скінченних елементах, що утворюють дискретну модель задачі. Для визначення значень шуканої неперервної величини у вузлах дискретної моделі будується система лінійних алгебраїчних рівнянь, матриця якої носить назву глобальної матриці жорсткості. Водночас вибір форми елемента і якість дискретизації вихідної області впливають на точність розв'язку. Так від форми скінченного елемента залежить степінь полінома, який використовується для апроксимації, а від якості дискретної моделі – значення якобіанів.

Типова схема дослідження поведінки об'єкта з використанням методу скінченних елементів ґрунтується на заміні варіаційної задачі системою лінійних алгебраїчних рівнянь вигляду:

$$[\mathbf{K}]\{\mathbf{X}\}=\{\mathbf{F}\}, \quad (2.1)$$

де $[\mathbf{K}]=\sum_e [\mathbf{K}_e]$ – глобальна матриця, яка є спеціальною сумою локальних

матриць, що апроксимують розв'язок на кожному скінченному елементі;

$\{\mathbf{X}\}$ – вектор, який апроксимує рішення задачі у вузлах скінченних елементів;

$\{\mathbf{F}\}$ – глобальний вектор, який є спеціальною сумою сил, які діють на об'єкт.

Інтерполяція $\{\mathbf{X}\}$ дозволяє обчислити значення шуканої величини в середині елементів. При побудові (2.1) враховують вимогу сумісності розв'язків на границях (гранях і ребрах) сусідніх елементів.

2.1.1 Двовимірні задачі дослідження напружено-деформованого стану

Для дослідження поведінки призматичних тіл або пластин, товщина яких є сталою, а зовнішні сили діють перпендикулярно до поздовжньої вісі та не змінюються уздовж товщини, часто використовують скінченно-елементні моделі плоско-напруженого або плоско-деформованого стану. У обох випадках поле переміщень однозначно визначається переміщеннями u і v , які відповідають осям координат x і y .

У результаті мінімізації енергії деформацій об'єкта, який знаходиться під дією зовнішніх сил, для локальних матриць жорсткості справедливим буде такий вираз:

$$[\mathbf{K}_e] = h \iint [\mathbf{B}]^T [\mathbf{D}] [\mathbf{B}] dx dy, \quad (2.2)$$

де h – товщина об'єкта;

$[\mathbf{B}]$ – матриця, яка пов'язує переміщення та деформації;

$[\mathbf{D}]$ – матриця, яка пов'язує деформації та напруження.

Матрицю $[\mathbf{B}]$ у (2.2) можна подати у такому вигляді:

$$[\mathbf{B}] = \begin{bmatrix} \frac{\partial N_1}{\partial x} & 0 & \frac{\partial N_2}{\partial x} & 0 & \dots & \frac{\partial N_m}{\partial x} & 0 \\ 0 & \frac{\partial N_1}{\partial y} & 0 & \frac{\partial N_2}{\partial y} & \dots & 0 & \frac{\partial N_m}{\partial y} \\ \frac{\partial N_1}{\partial y} & \frac{\partial N_1}{\partial x} & \frac{\partial N_2}{\partial y} & \frac{\partial N_2}{\partial x} & \dots & \frac{\partial N_m}{\partial y} & \frac{\partial N_m}{\partial x} \end{bmatrix}, \quad (2.3)$$

де $N_i = N_i(x, y)$ ($i = 1 \dots m$) – функції форми скінченного елемента;

m – кількість вузлів елемента.

Матриця $[\mathbf{D}]$ у (2.2) для ізотропного матеріалу з модулем Юнга E і коефіцієнтом Пуассона ν у випадку плоско-напруженого стану має такий вигляд [92]:

$$[\mathbf{D}] = \frac{E}{1 - \nu^2} \begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \frac{1 - \nu}{2} \end{bmatrix},$$

У випадку плоско-деформованого стану:

$$[\mathbf{D}] = \frac{E(1-\nu)}{(1+\nu)(1-2\nu)} \begin{bmatrix} 1 & \frac{\nu}{1-\nu} & 0 \\ \frac{\nu}{1-\nu} & 1 & 0 \\ 0 & 0 & \frac{1-2\nu}{2(1-\nu)} \end{bmatrix}.$$

Функції форми у (2.3) для лінійного трикутного скінченного елемента мають такий вигляд:

$$N_i(x, y) = \alpha_{i,0} + \alpha_{i,1}x + \alpha_{i,2}y, \quad i = \overline{1, 3},$$

а для білінійного чотирикутного скінченного елемента такий:

$$N_i(x, y) = \alpha_{i,0} + \alpha_{i,1}x + \alpha_{i,2}y + \alpha_{i,3}xy, \quad i = \overline{1, 4},$$

де $\alpha_{i,j}$ – коефіцієнти, які обирають таким чином, щоб функція форми i -го вузла була рівною одиниці у цьому вузлі та нулю в інших.

2.1.1.1 Згин консолі, навантаженої на кінці

Консоль, яка має вузький (щодо довжини) поперечний переріз одиничної товщини, згинається силою P , що прикладена на кінці (рис. 2.1). Ширина консолі дорівнює l , а висота $2c$.

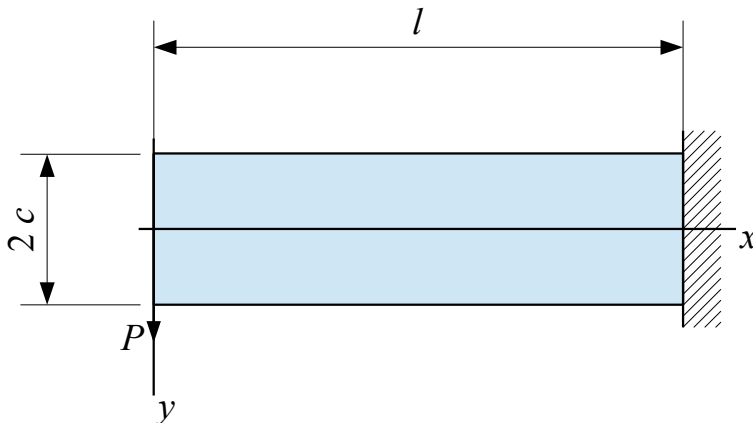


Рисунок 2.1 – Згин консолі: схема закріплення та дії сили

Якщо використовувати рівномірні дискретні моделі області, побудовані на 51×21 вузлі, то у результаті обчислювального експерименту при $P = 200$ МН, $E = 203200$ МПа, $\nu = 0,27$, $l = 10$ м, $c = 2$ м отримаємо, що при використанні чотирикутних елементів значення середини прогину навантаженого кінця консолі для середини консолі дорівнює $0,0635$ м (плоско-деформований стан). При використанні трикутних скінченних елементів – $0,0629$ м. Водночас відомо, що рівняння зігнутої осі консолі має такий вигляд [311]:

$$v(x)_{y=0} = \frac{Px^3}{6EI} - \frac{Pl^2x}{2EI} + \frac{Pl^3}{3EI} + \frac{Pc^2}{2IG}, \quad (2.4)$$

де $I = \frac{2}{3}c^3$ – момент інерції;

E – модуль пружності;

$G = \frac{E}{2(1+\nu)}$ – модуль зсуву;

ν – коефіцієнт Пуассона.

На кінці консолі при $x=0$ значення прогину, обчислене за формулою (2.4), дорівнює $0,0709$ м. Отже, похибка для максимального значення прогину осі консолі складає приблизно 10% при використанні чотирикутників і 11% – трикутників. На рис. 2.2 порівнюється чисельні та аналітичне значення прогину осі консолі.

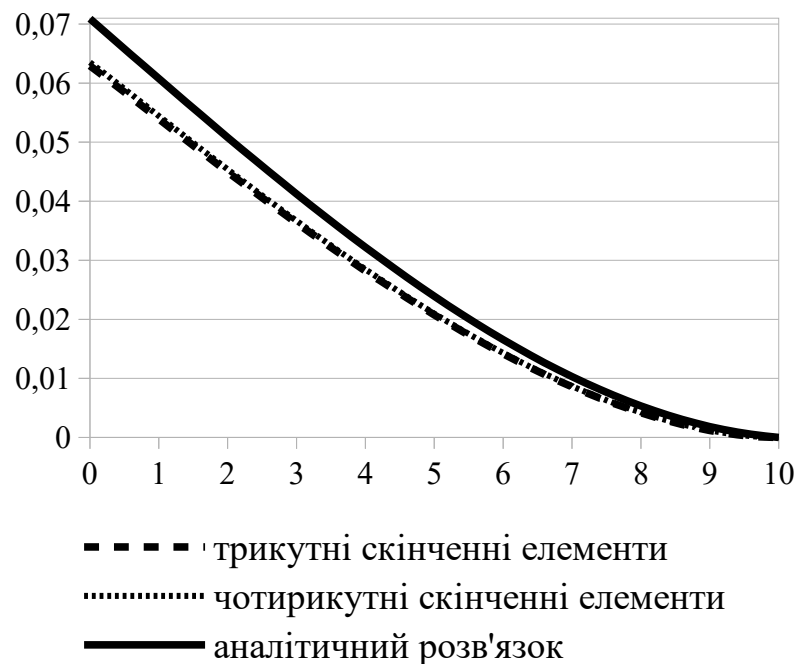


Рисунок 2.2 – Згин консолі: прогин осі консолі

Розподіл горизонтальних напружень у консолі описується формулою

$$\sigma_x = -\frac{P}{I} x y,$$

яка справедлива на значному віддаленні від кінців консолі [142]. Тому порівняємо значення напружень вздовж вертикальної осі консолі (при $x=l/2$). У табл. 2.1 наведено отримані результати тільки для інтервалу значень y від -2 до 0 (розподіл напружень по площині консолі є дзеркальним відносно початку координат).

Таблиця 2.1 – Порівняння розподілу напружень σ_x (МПа) вздовж вертикальної осі консолі

y , м	-2	-1,8	-1,6	-1,4	-1,2	-1	-0,8	-0,6	-0,4	-0,2	0
Трикутники	355,8	321,5	283,5	246,2	209,1	171,9	134,9	97,9	61,4	29,6	0
Чотирикутники	377,5	337,0	299,6	262,2	224,7	187,3	149,8	112,4	74,9	37,46	0
Аналітичний розв'язок	375	337,5	300	262,5	225	187,5	150	112,5	75	37,5	0
Трикутники (похибка, %)	5,13	4,75	5,50	6,21	7,09	8,30	10,09	12,97	18,20	20,96	–
Чотирикутники (похибка, %)	0,65	0,14	0,13	0,13	0,13	0,12	0,12	0,12	0,12	0,12	–

Отже, у результаті скінченно-елементного аналізу отримано, що чисельні значення для прогину осі консолі для трикутників і чотирикутників майже збігається, але останні дозволяють більш точно апроксимувати розподіл напружень, що обумовлено існуванням нелінійного члена у відповідних функціях форми.

2.1.1.2 Згин балки рівномірним навантаженням

Балка вузького прямокутного перерізу одиничної ширини, обперта на кінцях, згинається рівномірно розподіленим навантаженням q (рис. 2.3).

Якщо довжина балки дорівнює $2l$, її висота – $2c$, модуль пружності – E , а коефіцієнт Пуассона – ν , то крива прогинів її горизонтальної осі має вигляд [311]:

$$v(x)_{y=0} = \frac{5}{24} \frac{q l^4}{E I} \left[1 + \frac{12}{5} \frac{c^2}{l^2} \left(\frac{4}{5} + \frac{\nu}{2} \right) \right] - \frac{q}{2 E I} \left[\frac{l^2 x^2}{2} - \frac{x^4}{12} - \frac{1}{5} c^2 x^2 + \left(1 + \frac{1}{2} \nu \right) c^2 x^2 \right].$$

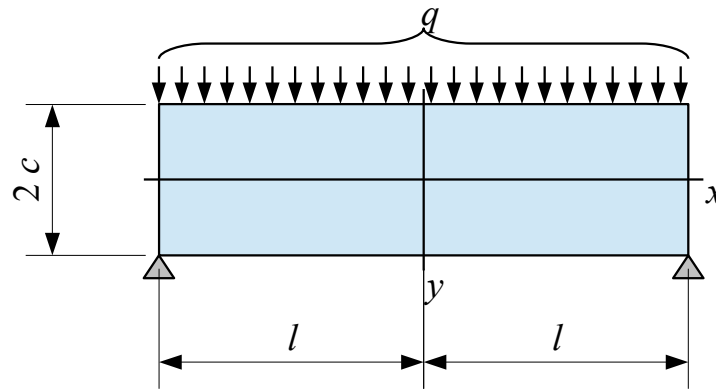


Рисунок 2.3 – Згин балки

Якщо для скінченно-елементного дослідження плоско-деформованого стану використовувати рівномірну структуровану дискретну модель, яка побудована на 51×11 вузлі для прямокутника $[-10; 10] \times [-2; 2]$ ($l=10$ м, $c=2$ м), то при інтенсивності навантаження $q=200$ МН/м, модулі пружності $E=203200$ МПа та коефіцієнті Пуассона $\nu=0,27$ похибка значень прогину складатиме у середньому 10% для трикутників і 7% для чотирикутників (рис. 2.4, а). Водночас на віддалені від кінців балки розподіл напружень σ_x визначається такою формулою [311]:

$$\sigma_x(x, y) = \frac{q}{2I} (l^2 - x^2) y + \frac{q}{2I} \left(\frac{2}{3} y^3 - \frac{2}{5} c^2 y \right).$$

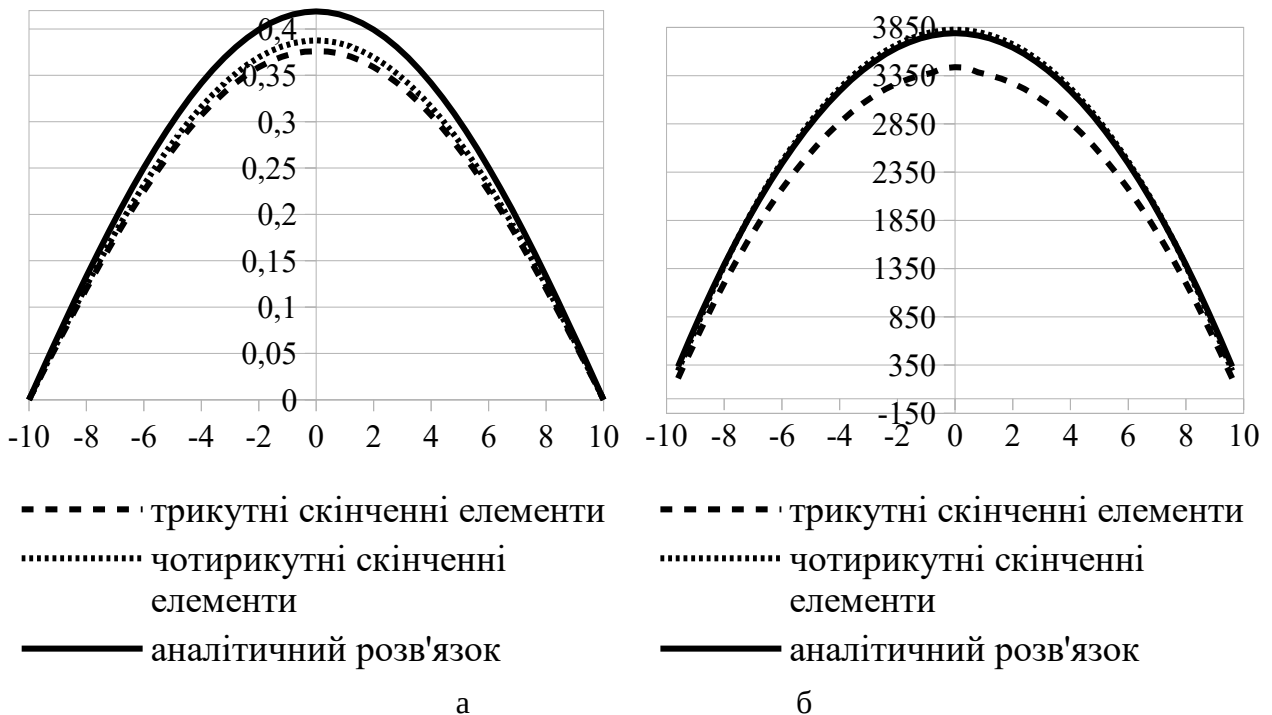


Рисунок 2.4 – Напружено-деформований стан балки: а – прогин горизонтальної осі (м); б – розподіл напружень σ_x вдовж горизонтальної осі (МПа)

На рис. 2.4, б можна побачити, що чотирикутні елементи більш точно апроксимують розподіл напружень уздовж горизонтальної осі балки.

2.1.2 Тривимірні задачі дослідження напружено-деформованого стану

У результаті мінімізації енергії деформацій об'єкта, який знаходиться під дією зовнішніх сил, у тривимірному випадку для локальних матриць жорсткості справедливим буде такий вираз:

$$[\mathbf{K}_e] = \iiint [\mathbf{B}]^T [\mathbf{D}] [\mathbf{B}] dx dy dz, \quad (2.5)$$

де $[\mathbf{B}]$ – матриця, яка пов'язує переміщення та деформації;

$[\mathbf{D}]$ – матриця, яка пов'язує деформації та напруження.

Матрицю $[\mathbf{B}]$ у (2.5) для випадку тривимірного напружено-деформованого стану можна подати у такому вигляді:

$$[\mathbf{B}] = \begin{bmatrix} \frac{\partial N_1}{\partial x} & 0 & 0 & \frac{\partial N_2}{\partial x} & 0 & 0 & \dots & \frac{\partial N_m}{\partial x} & 0 & 0 \\ 0 & \frac{\partial N_1}{\partial y} & 0 & 0 & \frac{\partial N_2}{\partial y} & 0 & \dots & 0 & \frac{\partial N_m}{\partial y} & 0 \\ 0 & 0 & \frac{\partial N_1}{\partial z} & 0 & 0 & \frac{\partial N_2}{\partial z} & \dots & 0 & 0 & \frac{\partial N_m}{\partial z} \\ \frac{\partial N_1}{\partial y} & \frac{\partial N_1}{\partial x} & 0 & \frac{\partial N_2}{\partial y} & \frac{\partial N_2}{\partial x} & 0 & \dots & \frac{\partial N_m}{\partial y} & \frac{\partial N_m}{\partial x} & 0 \\ 0 & \frac{\partial N_1}{\partial z} & \frac{\partial N_1}{\partial y} & 0 & \frac{\partial N_2}{\partial z} & \frac{\partial N_2}{\partial y} & \dots & 0 & \frac{\partial N_m}{\partial z} & \frac{\partial N_m}{\partial y} \\ \frac{\partial N_1}{\partial z} & 0 & \frac{\partial N_1}{\partial x} & \frac{\partial N_2}{\partial z} & 0 & \frac{\partial N_2}{\partial x} & \dots & \frac{\partial N_m}{\partial z} & 0 & \frac{\partial N_m}{\partial x} \end{bmatrix}, \quad (2.6)$$

де $N_i = N_i(x, y, z)$ ($i = 1 \dots m$) – функції форми скінченного елемента;

m – кількість вузлів елемента.

Матриця $[\mathbf{D}]$ у (2.5) для ізотропного матеріалу з модулем Юнга E і коефіцієнтом Пуассона ν у випадку тривимірного напружено-деформованого стану має такий вигляд [92]:

$$[D] = \frac{E(1-\nu)}{(1+\nu)(1-2\nu)} \begin{bmatrix} 1 & \frac{\nu}{1-\nu} & \frac{\nu}{1-\nu} & 0 & 0 & 0 \\ \frac{\nu}{1-\nu} & 1 & \frac{\nu}{1-\nu} & 0 & 0 & 0 \\ \frac{\nu}{1-\nu} & \frac{\nu}{1-\nu} & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1-2\nu}{2(1-\nu)} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1-2\nu}{2(1-\nu)} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1-2\nu}{2(1-\nu)} \end{bmatrix}.$$

Функції форми у (2.6) для лінійного тетраедричного скінченного елемента мають вигляд:

$$N_i(x, y, z) = \alpha_{i,0} + \alpha_{i,1}x + \alpha_{i,2}y + \alpha_{i,3}z, \quad i = \overline{1, 4},$$

Для трилінійного шестигранного скінченного елемента:

$$N_i(x, y, z) = \alpha_{i,0} + \alpha_{i,1}x + \alpha_{i,2}y + \alpha_{i,3}z + \alpha_{i,4}xy + \alpha_{i,5}xz + \alpha_{i,6}yz + \alpha_{i,7}xyz, \quad i \in \overline{1, 8}.$$

де $\alpha_{i,j}$ – коефіцієнти, які обирають таким чином, щоб функція форми i -го вузла дорівнювала одиниці у цьому вузлі та нулю в інших.

Розглянемо напружено-деформований стан призматичного стрижня під дією власної ваги. Якщо вертикальна вісь стрижня збігається з Oz, об'ємні сили діють у додатному напрямі цієї осі, $\gamma = \rho g$ – це питома вага стрижня, l – його довжина, E – модуль Юнга, ν – коефіцієнт Пуассона, то переміщення вздовж його вісі визначаються такою формулою [311]:

$$w(x, y, z) = \frac{\gamma z^2}{2E} + \frac{\nu \gamma}{2E} (x^2 + y^2) - \frac{\gamma l^2}{2E}. \quad (2.7)$$

Водночас у стрижні діють напруження

$$\sigma_z = \gamma z. \quad (2.8)$$

Для скінченно-елементного дослідження поведінки стрижня використаємо рівномірні структуровані дискретні моделі призматичної області $1 \times 1 \times 20$ м (довжина $l = 20$ м), які побудовані на $3 \times 3 \times 41$ вузлі (рис. 2.5). Питома

вага $\gamma=0,0765 \text{ МН/м}^3$, модуль Юнга $E=210000 \text{ МПа}$, коефіцієнт Пуассона $\nu=0,3$. Вважаємо, що для кінця стрижня $z=l$ заборонені переміщення у третьому напрямі (вздовж осі Oz), а об'ємна сила, яка дорівнює γ , діє додатному напрямку.

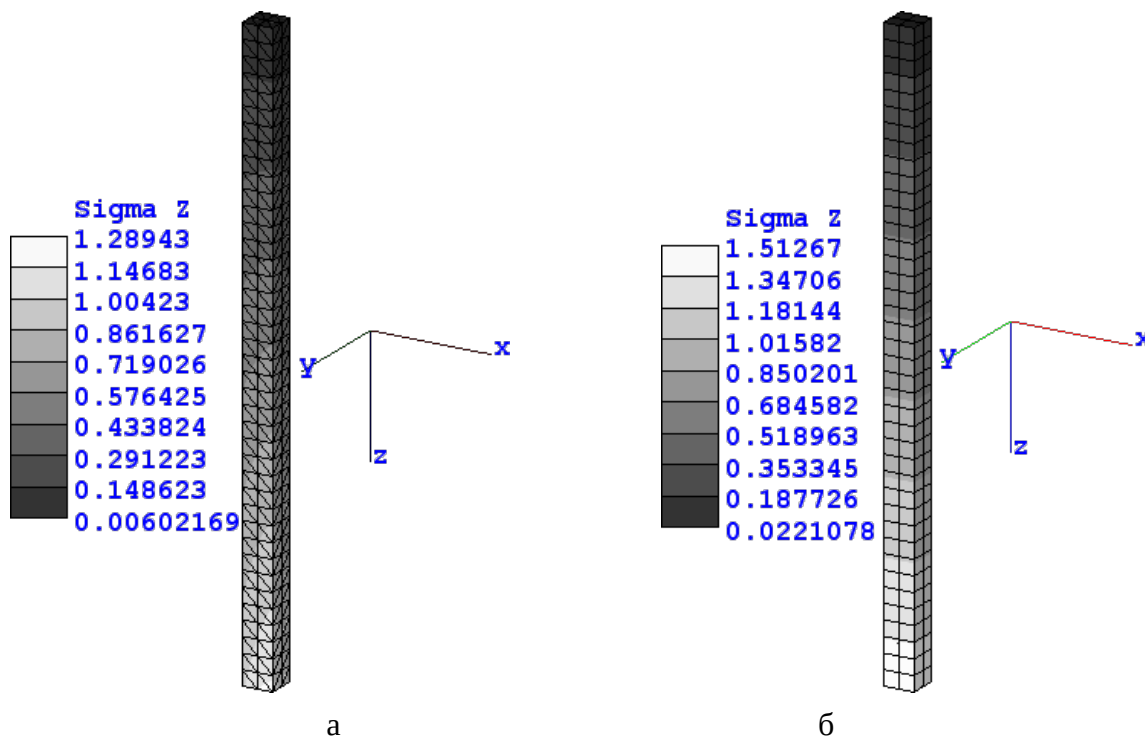


Рисунок 2.5 – Розподіл напружень σ_z (МПа) у дискретній моделі стрижня: а – тетраедричні скінченні елементи; б – шестигранні скінченні елементи

У результаті скінченно-елементного аналізу отримано, що при використанні тетраедрів, максимальне подовження осі стрижня під дією власної ваги становить $7,2867 \times 10^{-5} \text{ м}$, при використанні кубиків – $7,2866 \times 10^{-5} \text{ м}$. Водночас для осі стрижня (якщо $x=0$ і $y=0$) за формулою (2.7) максимальне подовження дорівнює $7,2857 \times 10^{-5} \text{ м}$. Отже, різниця між розв'язками складає менше одного відсотку (порівняння похибки відносно аналітичного розв'язку зображене на рис. 2.6, а). На рис. 2.6, б порівнюються розподіли значень напружень уздовж осі стрижня, отримані на базі кубічних і тетраедричних елементів, аналітичного розв'язку.

Можна побачити, що обидві форми елементів вельми точно апроксимують переміщення (похибка менша 1%), але шестигранники (кубики) більш точні в обчисленні напружень (розподіл для шестигранників майже збігається з аналітичним розв'язком). Це обумовлено тим, що функції форми шестигранних скінченних елементів (на відміну від тетраедричних) мають чотири нелінійних члени. У результаті для апроксимації похідних використовуються несталі вирази.

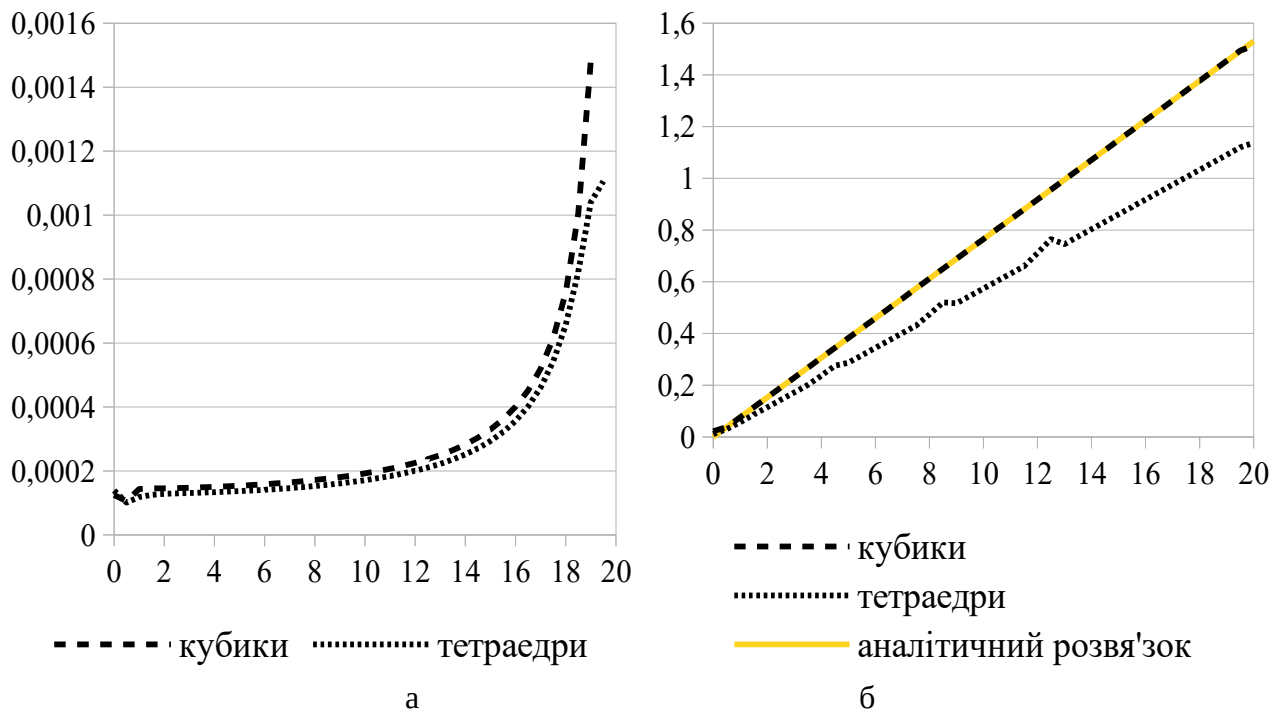


Рисунок 2.6 – Напружено-деформований стан стрижня: а – похибка переміщення вісі у третьому напрямі відносно аналітичного розв'язку; б – розподіл напружень уздовж осі стрижня (МПа)

Отже, за результатами виконаних обчислювальних експериментів можна зробити висновок, що, з чисельної точки зору, використання чотирикутників і шестигранників дозволяє отримати більш точні розв'язки відносно трикутників і тетраедрів.

2.2. Скінченно-елементне моделювання оболонок ракетної техніки

З точки зору математичного моделювання елементи ракет-носіїв раціонально подавати оболонками – тривимірними тілами, у яких один розмір (товщина) значно менший за інші. Такий підхід обумовлений раціональністю використання оболонкових скінченно-елементних моделей, які, зазвичай, мають значно меншу кількість елементів відносно тривимірних моделей на базі елементів типу “солід”, а ще й можливістю підвищення точності шляхом більш точного врахування механічних особливостей поведінки таких конструкцій.

При розробці підходів до моделювання оболонок ракетної техніки необхідно враховувати, що такі конструкції, зазвичай, знаходяться під дією навантажень, які зумовлюють пластичні деформації. Також у процесі експлуатації в елементах оболонок ракетної техніки додатково виникають температурні деформації.

2.2.1 Моделювання тонких і середніх оболонок як сукупності плоских елементів із ізотропного матеріалу

Дво- і тривимірні ізопериметричні елементи заслужено вважають вельми ефективними в аналізі напружено-деформованого стану суцільних тіл. Здавалося б, що ці елементи можна безпосередньо застосовувати при розрахунку оболонкових конструкцій, зменшуючи їх розмір в напрямку товщини оболонки. Проте у загальному випадку застосування таких скінченних елементів для моделювання оболонок пов'язане з певними труднощами [93].

По-перше, наявність трьох ступенів свободи в кожному вузлі призводить до великих коефіцієнтів жорсткості для переміщень по товщині оболонки [93], що може бути чинником поганої обумовленості системи рівнянь, якщо товщина оболонки є малою величиною в порівнянні з іншими розмірами скінченного елемента. По-друге, використання декількох вузлів по товщині оболонки є надлишковим. Це пояснює відомий факт, що в ряді випадків для товстих оболонок нормалі до серединних поверхонь можна вважати прямими.

Можливим способом вирішення цих питань є розробка моделі напружено-деформованого стану оболонкової конструкції, в якій оболонку представляють як сукупність плоских елементів. Водночас передбачають, що поведінка неперервної криволінійної поверхні досить точно характеризує поведінку поверхні, складеної з малих плоских елементів. З фізичних міркувань випливає, що зі зменшенням розмірів елемента рішення повинно сходитися і, як свідчить досвід, така збіжність дійсно спостерігається [93].

Нехай діють стандартні гіпотези про малість переміщень і деформацій, а також про відсутність стиснення і збереження нормалей серединної поверхні [312]. При моделюванні оболонок для кожного скінченного елемента локальну систему координат обрано так, щоб площина xOy збігалася з площиною елемента, вісь Oz була паралельна нормалі, а початок координат поєднувався з першим вузлом (рис. 2.7).

Деформований стан скінченного елемента для кожного i -го вузла однозначно описують переміщеннями u_i та v_i , які виникають під дією мембранних сил, прогином w_i та трьома кутами повороту нормалі: θ_x , θ_y і θ_z . Водночас потрібно врахувати дві обставини. По-перше, переміщення, викликані мембранними силами, та деформації згину не залежать одини від одного. По-друге, кут повороту θ_z , який у локальній системі координат не входить до числа параметрів, що визначають напружено-деформований стан, введено як фіктивний поворот нормалі навколо своєї вісі для забезпечення ансамблю елементів. Водночас вважають, що деформації згину відповідають

положенням теорії Рейсснера-Міндліна [312, 313] (теорія зсувних деформацій другого порядку для тонких і середніх пластин і оболонок). У результаті поле передбачуваних переміщень визначається такими співвідношеннями:

$$\begin{cases} u(x, y, z) = u(x, y) + z \theta_x(x, y), \\ v(x, y, z) = v(x, y) + z \theta_y(x, y), \\ w(x, y, z) = w(x, y). \end{cases} \quad (2.9)$$

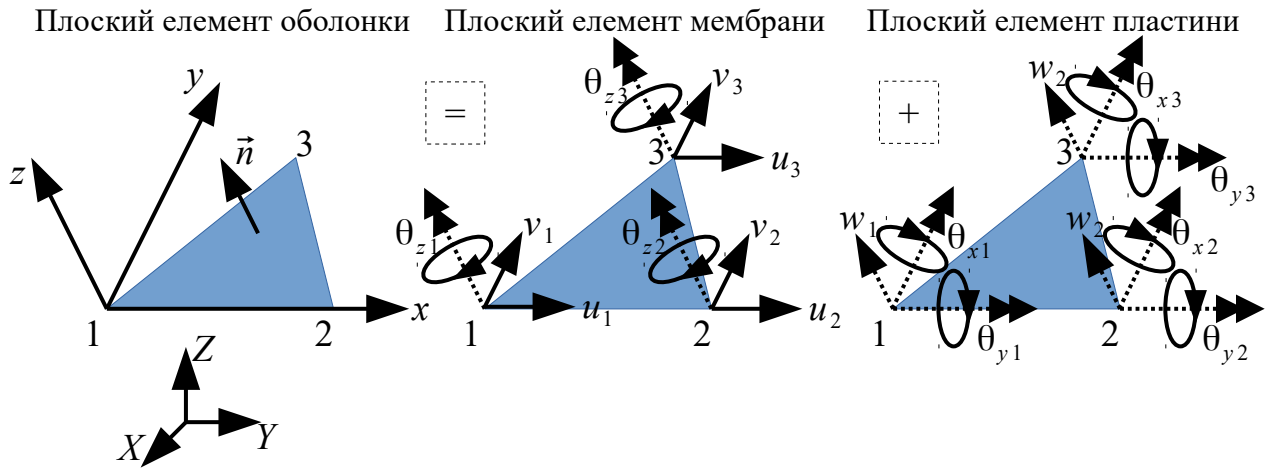


Рисунок 2.7 – Степені свободи елемента оболонки

Вектор узагальнених переміщень у вузлах скінченного елемента набуває такого вигляду:

$$\{\mathbf{d}_e\}^T = \{u_1 \ v_1 \ w_1 \ \theta_{x1} \ \theta_{y1} \ \theta_{z1} \ \dots \ u_{n_e} \ v_{n_e} \ w_{n_e} \ \theta_{xn_e} \ \theta_{yn_e} \ \theta_{zn_e}\} \quad (2.10)$$

де n_e – кількість вузлів у елементі ($n_e=3$ для трикутників і $n_e=4$ для чотирикутників).

Вектор плоских деформацій можна визначити диференціюванням функцій переміщень:

$$\{\boldsymbol{\varepsilon}\} = \begin{Bmatrix} \varepsilon_x \\ \varepsilon_y \\ \gamma_{xy} \end{Bmatrix} = \{\boldsymbol{\varepsilon}_m\} + z \{\boldsymbol{\varepsilon}_b\} = \begin{Bmatrix} \varepsilon_x^{(m)} \\ \varepsilon_y^{(m)} \\ \gamma_{xy}^{(m)} \end{Bmatrix} + z \begin{Bmatrix} \varepsilon_x^{(b)} \\ \varepsilon_y^{(b)} \\ \gamma_{xy}^{(b)} \end{Bmatrix} = \begin{Bmatrix} u_{,x} \\ v_{,y} \\ u_{,y} + v_{,x} \end{Bmatrix} + z \begin{Bmatrix} \theta_{x,x} \\ \theta_{y,y} \\ \theta_{x,y} + \theta_{y,x} \end{Bmatrix}, \quad (2.11)$$

де $\{\boldsymbol{\varepsilon}_m\}$ – вектор мембранних деформацій;

$\{\boldsymbol{\varepsilon}_b\}$ – вектор деформацій згину;

$$u_{,x} = \frac{\partial u}{\partial x}, \quad u_{,y} = \frac{\partial u}{\partial y}, \quad v_{,x} = \frac{\partial v}{\partial x}, \quad v_{,y} = \frac{\partial v}{\partial y}, \quad \theta_{x,x} = \frac{\partial \theta_x}{\partial x}, \quad \theta_{x,y} = \frac{\partial \theta_x}{\partial y}, \quad \theta_{y,x} = \frac{\partial \theta_y}{\partial x} \text{ і}$$

$\theta_{y,y} = \frac{\partial \theta_y}{\partial y}$ – похідні переміщень.

Вектор деформацій зсуву можна отримати у такому вигляді:

$$\{\boldsymbol{\gamma}\} = \begin{Bmatrix} \gamma_{xz} \\ \gamma_{yz} \end{Bmatrix} = \begin{Bmatrix} w_{,x} + \theta_x \\ w_{,y} + \theta_y \end{Bmatrix}, \quad (2.12)$$

де $w_{,x} = \frac{\partial w}{\partial x}$ і $w_{,y} = \frac{\partial w}{\partial y}$ – похідні прогину.

Співвідношення між напруженнями та деформаціями для ізотропного матеріалу можна записати у такому вигляді:

$$\{\boldsymbol{\sigma}\} = \begin{Bmatrix} \sigma_x \\ \sigma_y \\ \tau_{xy} \end{Bmatrix} = [\mathbf{D}] \boldsymbol{\varepsilon} = \frac{E}{1-\mu^2} \begin{bmatrix} 1 & \mu & 0 \\ \mu & 1 & 0 \\ 0 & 0 & \frac{1-\mu}{2} \end{bmatrix} \begin{Bmatrix} \varepsilon_x \\ \varepsilon_y \\ \gamma_{xy} \end{Bmatrix}, \quad (2.13)$$

$$\{\boldsymbol{\tau}\} = \begin{Bmatrix} \tau_{xz} \\ \tau_{yz} \end{Bmatrix} = [\mathbf{G}] \{\boldsymbol{\gamma}\} = \begin{bmatrix} G & 0 \\ 0 & G \end{bmatrix} \begin{Bmatrix} \gamma_{xz} \\ \gamma_{yz} \end{Bmatrix}, \quad (2.14)$$

де E – модуль Юнга;

μ – коефіцієнт Пуассона;

$G = \frac{E}{2(1+\mu)}$ – модуль зсуву.

Енергію деформації елементарного об'єму V_e ізотропної оболонки, яка знаходиться під дією зовнішніх сил, можна подати у такому вигляді:

$$\Pi = \frac{1}{2} \int_{V_e} \{\boldsymbol{\sigma}\}^T \{\boldsymbol{\varepsilon}\} dV_e + \frac{\kappa}{2} \int_{V_e} \{\boldsymbol{\tau}\}^T \{\boldsymbol{\gamma}\} dV - \mathbf{A} \quad (2.15)$$

де $\kappa = \frac{5}{6}$ – коефіцієнт корекції енергії зсуву;

$\mathbf{A}$ – робота зовнішніх сил.

Узагальнені переміщення можна отримати шляхом інтерполяції у вузлах скінченного елемента:

$$u = \sum_{i=0}^{n_e} N_i(\xi, \eta) u_i, \quad (2.16)$$

$$v = \sum_{i=0}^{n_e} N_i(\xi, \eta) v_i, \quad (2.17)$$

$$w = \sum_{i=0}^{n_e} N_i(\xi, \eta) w_i, \quad (2.18)$$

$$\theta_x = \sum_{i=0}^{n_e} N_i(\xi, \eta) \theta_{xi}, \quad (2.19)$$

$$\theta_y = \sum_{i=0}^{n_e} N_i(\xi, \eta) \theta_{yi}, \quad (2.20)$$

де n_e – кількість вузлів у скінченному елементі.

У результаті вставки (2.11)–(2.14) і (2.16)–(2.20) у (2.15), якщо на поверхню оболонки діє навантаження, задане функцією q , отримаємо таке співвідношення для енергії:

$$\begin{aligned} \Pi = & \frac{1}{2} \int_{V_e} ([\mathbf{B}_m] \{\mathbf{d}_e\} + z [\mathbf{B}_b] \{\mathbf{d}_e\})^T [\mathbf{D}] ([\mathbf{B}_m] \{\mathbf{d}_e\} + z [\mathbf{B}_b] \{\mathbf{d}_e\}) dV_e + \\ & + \frac{\kappa}{2} \int_{V_e} ([\mathbf{B}_c] \{\mathbf{d}_e\})^T [\mathbf{G}] ([\mathbf{B}_c] \{\mathbf{d}_e\}) dV_e - \int_{S_e} (\{\mathbf{N}\}^T \{\mathbf{d}_e\}) q dS_e, \end{aligned} \quad (2.21)$$

де $[\mathbf{B}_m]$, $[\mathbf{B}_b]$ і $[\mathbf{B}_c]$ – матриці зв'язків між узагальненими переміщеннями та, відповідно, мембранними деформаціями, деформаціями прогину та зсуву;

V_e – елементарний об'єм оболонки;

S_e – площа серединної поверхні елементарного об'єму оболонки;

$\{\mathbf{N}\}^T = \{N_1 \ N_2 \ \dots \ N_{n_e}\}$ – вектор-стовпець функцій форми.

У результаті мінімізації (2.21) отримаємо

$$\begin{aligned} \int_{V_e} ([\mathbf{B}_m]^T [\mathbf{D}] [\mathbf{B}_m] + z [\mathbf{B}_m]^T [\mathbf{D}] [\mathbf{B}_b] + z [\mathbf{B}_b]^T [\mathbf{D}] [\mathbf{B}_m] + z^2 [\mathbf{B}_b]^T [\mathbf{D}] [\mathbf{B}_b]) dV_e + \\ + \kappa \int_{V_e} [\mathbf{B}_c]^T [\mathbf{G}] [\mathbf{B}_c] dV_e = \int_{S_e} \{\mathbf{N}\} q dS_e. \end{aligned} \quad (2.22)$$

Після інтегрування лівої частини рівняння (2.22) за товщиною елемента оболонки (у межах від $-h/2$ до $h/2$) отримаємо таку формулу для локальної матриці жорсткості елемента оболонки із ізотропного матеріалу:

$$[\mathbf{K}_e^{\text{local}}] = h \int_{S_e} [\mathbf{B}_m]^T [\mathbf{D}] [\mathbf{B}_m] dS_e + \frac{h^3}{12} \int_{S_e} [\mathbf{B}_b]^T [\mathbf{D}] [\mathbf{B}_b] dS_e + \kappa h \int_{S_e} [\mathbf{B}_c]^T [\mathbf{G}] [\mathbf{B}_c] dS_e. \quad (2.23)$$

Матриці зв'язків між переміщеннями та деформаціями для вектора узагальнених переміщень (2.10) у формулах (2.21)–(2.23) отримають такий вигляд:

$$[\mathbf{B}_m] = \begin{bmatrix} \frac{\partial N_1}{\partial x} & 0 & 0 & 0 & 0 & 0 & \dots & \frac{\partial N_{n_e}}{\partial x} & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{\partial N_1}{\partial y} & 0 & 0 & 0 & 0 & \dots & 0 & \frac{\partial N_{n_e}}{\partial y} & 0 & 0 & 0 & 0 \\ \frac{\partial N_1}{\partial y} & \frac{\partial N_1}{\partial x} & 0 & 0 & 0 & 0 & \dots & \frac{\partial N_{n_e}}{\partial y} & \frac{\partial N_{n_e}}{\partial x} & 0 & 0 & 0 & 0 \end{bmatrix}, \quad (2.24)$$

$$[\mathbf{B}_b] = \begin{bmatrix} 0 & 0 & 0 & \frac{\partial N_1}{\partial x} & 0 & 0 & \dots & 0 & 0 & 0 & \frac{\partial N_{n_e}}{\partial x} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{\partial N_1}{\partial y} & 0 & \dots & 0 & 0 & 0 & 0 & \frac{\partial N_{n_e}}{\partial y} & 0 \\ 0 & 0 & 0 & \frac{\partial N_1}{\partial y} & \frac{\partial N_1}{\partial x} & 0 & \dots & 0 & 0 & 0 & \frac{\partial N_{n_e}}{\partial y} & \frac{\partial N_{n_e}}{\partial x} & 0 \end{bmatrix}, \quad (2.25)$$

$$\mathbf{B}_c = \begin{bmatrix} 0 & 0 & \frac{\partial N_1}{\partial x} & N_1 & 0 & 0 & \dots & 0 & 0 & \frac{\partial N_{n_e}}{\partial x} & N_{n_e} & 0 & 0 \\ 0 & 0 & \frac{\partial N_1}{\partial y} & 0 & N_1 & 0 & \dots & 0 & 0 & \frac{\partial N_{n_e}}{\partial y} & 0 & N_{n_e} & 0 \end{bmatrix}. \quad (2.26)$$

Після інтегрування (2.23) за площиною серединної поверхні елемента отримаємо локальну матрицю жорсткості (розмір матриці: $6n_e \times 6n_e$) у місцевій системі координат. Для обчислення глобальної матриці жорсткості у глобальній системі координат потрібно виконати таке перетворення:

$$[\mathbf{K}_e^{\text{global}}] = [\mathbf{R}]^T [\mathbf{K}_e^{\text{local}}] [\mathbf{R}], \quad (2.27)$$

де

$$[\mathbf{R}] = \begin{bmatrix} [\mathbf{T}_{6 \times 6}] & [\mathbf{0}_{6 \times 6}] & \dots & [\mathbf{0}_{6 \times 6}] \\ [\mathbf{0}_{6 \times 6}] & [\mathbf{T}_{6 \times 6}] & \dots & [\mathbf{0}_{6 \times 6}] \\ \dots & \dots & \dots & \dots \\ [\mathbf{0}_{6 \times 6}] & [\mathbf{0}_{6 \times 6}] & \dots & [\mathbf{T}_{6 \times 6}] \end{bmatrix}, \quad (2.28)$$

$$[\mathbf{T}_{6 \times 6}] = \begin{bmatrix} c_{11} & c_{12} & c_{13} & 0 & 0 & 0 \\ c_{21} & c_{22} & c_{23} & 0 & 0 & 0 \\ c_{31} & c_{32} & c_{33} & 0 & 0 & 0 \\ 0 & 0 & 0 & c_{11} & c_{12} & c_{13} \\ 0 & 0 & 0 & c_{21} & c_{22} & c_{23} \\ 0 & 0 & 0 & c_{31} & c_{32} & c_{33} \end{bmatrix}, \quad (2.29)$$

$$[0_{6 \times 6}] = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

У формулі (2.29) c_{ij} – направляючі косинуси перетворення глобальної системи координат у локальну систему координат елементу.

Як було зазначено вище кут повороту θ_z не входить до числа параметрів, що визначають напружено-деформований стан у локальній системі координат. Це може зумовити сингулярність глобальної матриці жорсткості у випадках, якщо оболонка має малу кривину (близька до площини) [93]. Найпростіший спосіб усунення такої сингулярності – додавання малої фіктивної жорсткості у позиціях матриці $[\mathbf{K}_e^{\text{local}}]$, які відповідають θ_z . Наприклад, у таких позиціях можна встановлювати значення $10^{-3} \max([\mathbf{E}][\mathbf{K}_e^{\text{local}}])$ [314].

2.2.1.1 Дослідження прогину циліндричної оболонки

Для оцінки точності розробленої скінченно-елементної моделі розглянемо задачу обчислення прогину циліндричної оболонки з вільно обпертими торцями, яка знаходиться під дією рівномірного внутрішнього тиску (рис. 2.8).

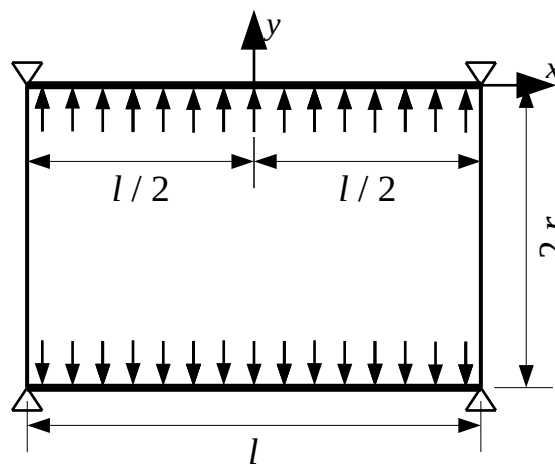


Рисунок 2.8 – Схема обчислювального експерименту

Прогин оболонки у будь-якій точці оболонки можна обчислити за допомогою відомої аналітичної формули:

$$w(x) = \frac{-p l^4}{64 D \alpha^4} \left(1 - \frac{2 \sin \alpha \sinh \alpha}{\cos(2\alpha) + \cosh(2\alpha)} \sin(\beta x) \sinh(\beta x) - \frac{2 \cos \alpha \cosh \alpha}{\cos(2\alpha) + \cosh(2\alpha)} \cos(\beta x) \cos(\beta x) \right),$$

$$\text{де } \alpha = \frac{\beta l}{2};$$

$$\beta = \sqrt[4]{\frac{3(1-\nu^2)}{r^2 h^2}};$$

$$D = \frac{E h^3}{12(1-\nu^2)};$$

l – довжина оболонки;

h – товщина оболонки;

r – радіус оболонки;

E – модуль Юнга;

ν – коефіцієнт Пуассона;

p – значення внутрішнього тиску.

На рис. 2.9 порівняно аналітичний та скінченно-елементі розв'язки.

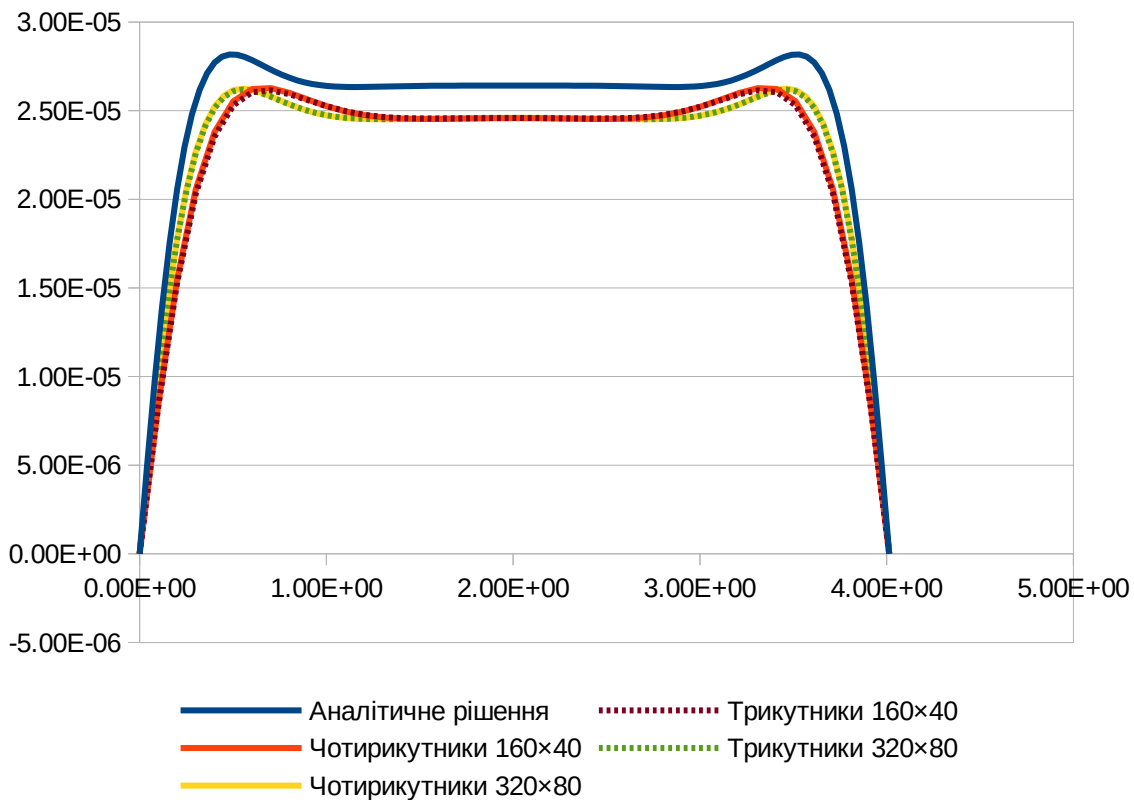


Рисунок 2.9 – Порівняння значень прогину оболонки

2.2.1.2 Дослідження напружено-деформованого стану баку

Як інший приклад розглянемо задачу дослідження напружено-деформованого стану паливного баку ракети-носія. Бак – циліндрична конструкція, яка складається з більшої частини, обмеженої двома сферичними днищами, що містить паливо, і меншої частини, що містить окиснювач (рис. 2.10).

Товщина оболонок баку – 0,0025 м, радіус циліндричної частини – 1,5 м, радіуси сферичних днищ – 2 м, довжина твірної циліндричної частини для палива – 4,165 м, довжина твірної циліндричної частини для окиснювача – 2,637 м, радіус труби, що з'єднує частини – 0,2 м. На правому днищі частини, що містить паливо є торцевий шпангоут з довжиною твірної – 0,06 м.

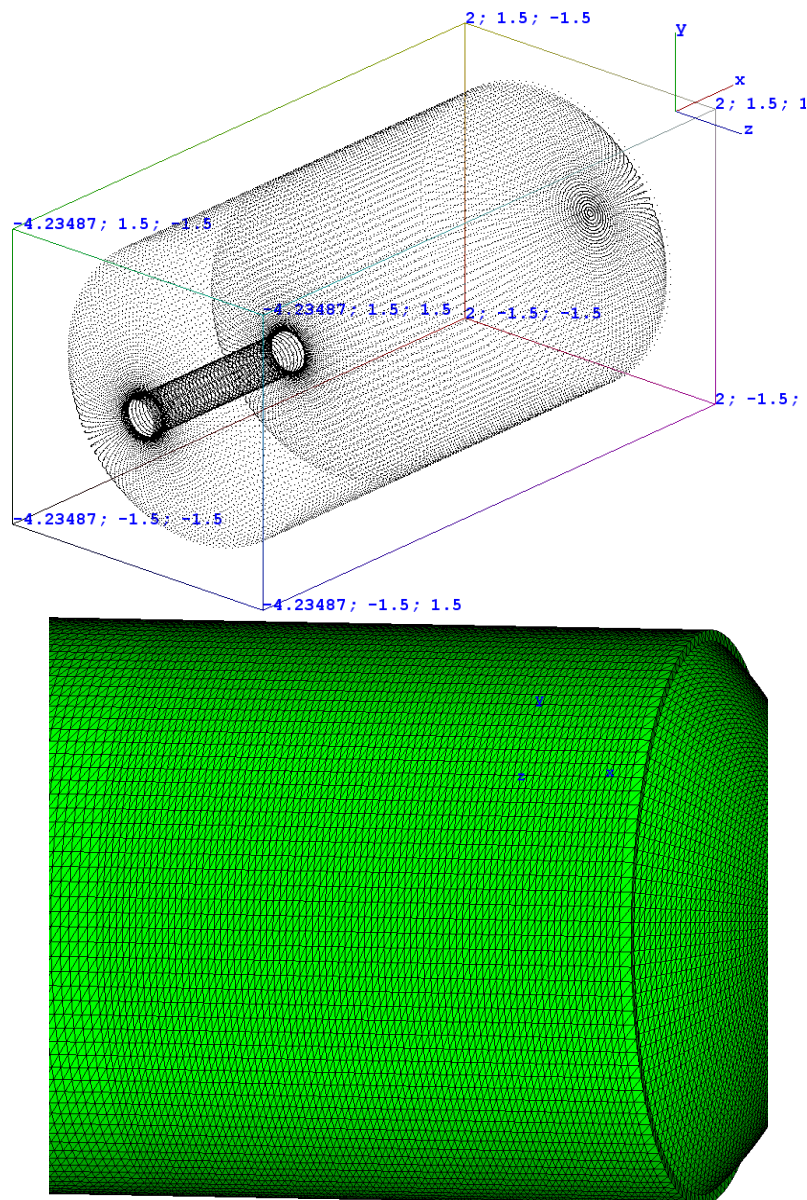


Рисунок 2.10 – Дискретна модель форми бака

Нехай бак виконано з алюмінієвого сплаву, у якого модуль Юнга $E=65000$ МПа і коефіцієнт Пуассона $\nu=0,3$, на паливний відсік діє внутрішній тиск з інтенсивністю $q=0,01$ МПа. У результаті скінченно-елементного аналізу на базі формули (2.23) встановлено, що максимальна інтенсивність напружень за Мізесом дорівнює 9,34 МПа. Область, на якій досягається максимальна інтенсивність напружень, знаходиться на обечайці лівіше шпангоута (рис. 2.11).

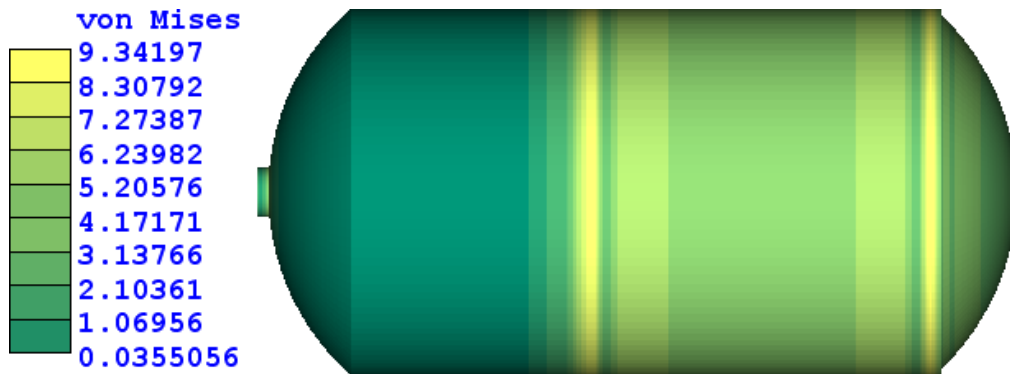


Рисунок 2.11 – Інтенсивність напружень за Мізесом, МПа

Якщо вважати, що межа міцності алюмінієвого сплаву дорівнює 330 МПа, то за допомогою лінійної інтерполяції можна оцінити значення руйнівного навантаження значенням 0,35 МПа.

2.2.2 Оболонки зі змінною товщиною

Скінченно-елементну модель, визначену формулами (2.23)–(2.29), отримано з припущенням, що оболонка має постійну товщину. Проте у багатьох практичних випадках товщина оболонки конструкції може змінюватися для забезпечення раціонального проектування.

Нехай товщину оболонки визначено кусково-постійною функцією $h(x, y, z)$. Водночас товщина оболонки в місцевих координатах залежить лише від x і y , а серединна поверхня оболонки є поверхнею, форма якої не залежить від функції $h(x, y, z)$. У такому випадку формула (2.23) матиме такого вигляду:

$$[\mathbf{K}_e^{\text{local}}] = \int_{S_e} h(x, y, z) [\mathbf{B}_m]^T [\mathbf{D}] [\mathbf{B}_m] dS_e + \frac{1}{12} \int_{S_e} h^3(x, y, z) [\mathbf{B}_b]^T [\mathbf{D}] [\mathbf{B}_b] dS_e + \kappa \int_{S_e} h(x, y, z) [\mathbf{B}_c]^T [\mathbf{G}] [\mathbf{B}_c] dS_e.$$

Наприклад, для дослідження паливного баку можна вважати, що товщина збільшується вдвічі у околах стиків днищ з циліндричною частиною з паливом (рис. 2.12).



Рисунок 2.12 – Зони зміни товщини баку

2.2.3 Температурні деформації в оболонках

Нехай на оболонку одночасно діють сили і температурне поле. Можна вважати, що поле переміщень з вектором узагальнених деформацій мають вигляд, відповідно, (2.9) і (2.10). У такому випадку вектор плоских деформацій (2.11) отримає такий вигляд:

$$\begin{aligned} \{\boldsymbol{\varepsilon}\} &= \begin{Bmatrix} \varepsilon_x \\ \varepsilon_y \\ \gamma_{xy} \end{Bmatrix} = \{\boldsymbol{\varepsilon}_m\} + z\{\boldsymbol{\varepsilon}_b\} - \{\boldsymbol{\varepsilon}_t\} = \begin{Bmatrix} \varepsilon_x^{(m)} \\ \varepsilon_y^{(m)} \\ \gamma_{xy}^{(m)} \end{Bmatrix} + z \begin{Bmatrix} \varepsilon_x^{(b)} \\ \varepsilon_y^{(b)} \\ \gamma_{xy}^{(b)} \end{Bmatrix} - \begin{Bmatrix} \varepsilon_x^{(t)} \\ \varepsilon_y^{(t)} \\ 0 \end{Bmatrix} = \\ &= \begin{Bmatrix} u_{,x} \\ v_{,y} \\ u_{,y} + v_{,x} \end{Bmatrix} + z \begin{Bmatrix} \theta_{x,x} \\ \theta_{y,y} \\ \theta_{x,y} + \theta_{y,x} \end{Bmatrix} - \begin{Bmatrix} \alpha T \\ \alpha T \\ 0 \end{Bmatrix}, \end{aligned} \quad (2.30)$$

де α – коефіцієнт теплового розширення матеріалу;

T – температура.

У результаті підстановки (2.30) у (2.13) і (2.15) з апроксимаційними формулами (2.16)–(2.20) можна отримати вираз для потенційної енергії оболонки, яка знаходиться під дією тиску з інтенсивністю q , у такому вигляді:

$$\begin{aligned} \Pi &= \frac{1}{2} \int_{V_e} ([\mathbf{B}_m]\{\mathbf{d}_e\} + z[\mathbf{B}_b]\{\mathbf{d}_e\} - \boldsymbol{\varepsilon}_t)^T [\mathbf{D}] ([\mathbf{B}_m]\{\mathbf{d}_e\} + z[\mathbf{B}_b]\{\mathbf{d}_e\} - \boldsymbol{\varepsilon}_t) dV_e + \\ &+ \frac{\kappa}{2} \int_{V_e} ([\mathbf{B}_c]\{\mathbf{d}_e\})^T [\mathbf{G}] ([\mathbf{B}_c]\{\mathbf{d}_e\}) dV_e - \int_{S_e} (\{\mathbf{N}\}^T \{\mathbf{d}_e\}) q dS_e, \end{aligned} \quad (2.31)$$

де позначення аналогічні до введених у формулі (2.21).

У результаті мінімізації (2.31) можна отримати таку рівність:

$$\int_{V_e} \left([\mathbf{B}_m]^T [\mathbf{D}] [\mathbf{B}_m] + z [\mathbf{B}_m]^T [\mathbf{D}] [\mathbf{B}_b] + z [\mathbf{B}_b]^T [\mathbf{D}] [\mathbf{B}_m] + z^2 [\mathbf{B}_b]^T [\mathbf{D}] [\mathbf{B}_b] \right) dV_e + \kappa \int_{V_e} [\mathbf{B}_c]^T [\mathbf{G}] [\mathbf{B}_c] dV_e = \int_{S_e} \{\mathbf{N}\} q dS_e + \int_{V_e} [\mathbf{B}_m]^T [\mathbf{D}] \{\boldsymbol{\epsilon}_t\} dV_e + \int_{V_e} z [\mathbf{B}_b]^T [\mathbf{D}] \{\boldsymbol{\epsilon}_t\} dV_e. \quad (2.32)$$

Після інтегрування об'ємних інтегралів у формулі (2.32) за товщиною елемента оболонки отримаємо співвідношення (2.23)–(2.27) для локальної і глобальної матриць жорсткості, а також таку формулу для вектора навантажень у локальній системі координат:

$$\{\mathbf{F}_e^{\text{local}}\} = \int_{S_e} \{\mathbf{N}\} q dS_e + h \int_{S_e} [\mathbf{B}_m]^T [\mathbf{D}] \{\boldsymbol{\epsilon}_t\} dS_e. \quad (2.33)$$

У глобальній системі координат (2.33) матиме такий вигляд:

$$\{\mathbf{F}_e^{\text{global}}\} = [\mathbf{R}]^T \{\mathbf{F}_e^{\text{local}}\}. \quad (2.34)$$

У формулі (2.34) матрицю $[\mathbf{R}]$ визначено формулами (2.28) і (2.29).

Як практичний приклад розглянемо задачу дослідження напружено-деформованого стану паливного баку за умови дії температурних деформацій, викликаних нагрівом на 1° . Коефіцієнт теплового розширення алюмінієвого сплаву $\alpha = 24,7 \times 10^{-6}$. Якщо вважати, що бак затиснений на кінцівці труби, що з'єднує частини з паливом і окиснювачем, то можна отримати картини переміщень, представлені на рис. 2.13–2.16.

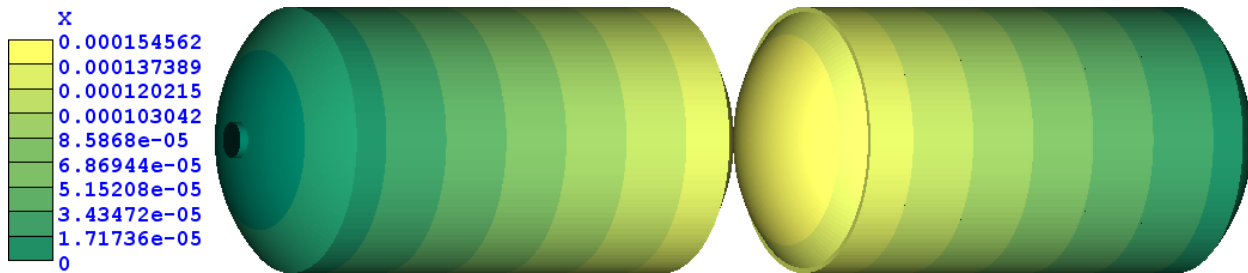


Рисунок 2.13 – Переміщення у першому напрямі, м

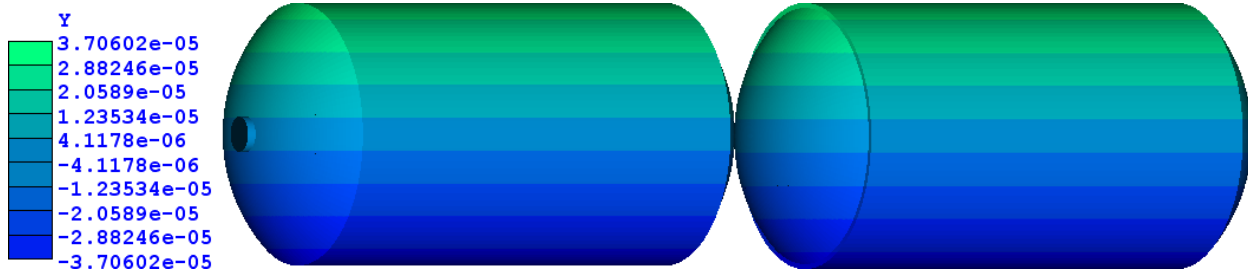


Рисунок 2.14 – Переміщення у другому напрямі, м

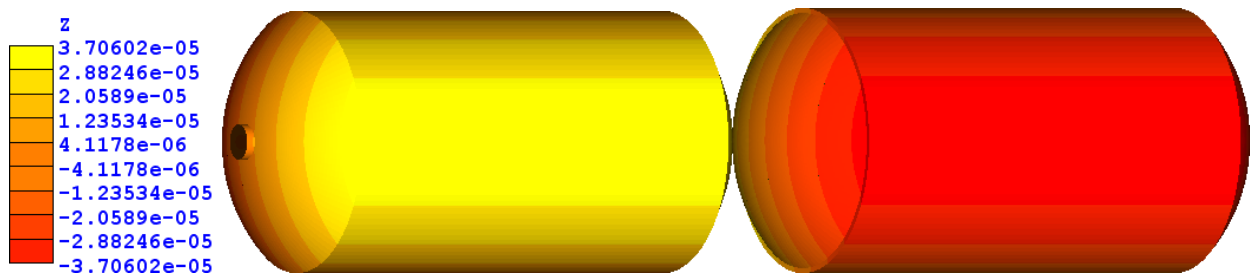


Рисунок 2.15 – Переміщення у третьому напрямі, м

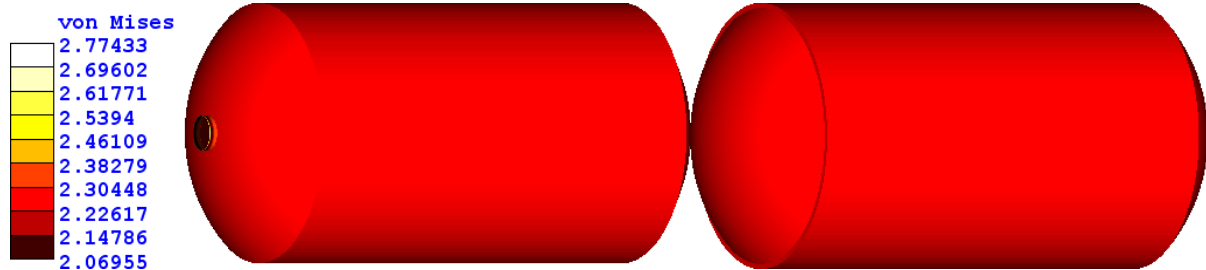


Рисунок 2.16 – Інтенсивність напружень за Мізесом, МПа

З рисунку 2.16 можна побачити, що нагрівання на 1° спричиняє збільшення інтенсивності напружень на 2–2,7 МПа.

2.2.4 Пружно-пластичний випадок на базі методу змінної пружності

Скінченно-елементне дослідження поведінки конструкцій на базі лінійної теорії пружності, зазвичай, перетворюють до розв'язання системи лінійних алгебраїчних рівнянь вигляду (2.1). Водночас для отримання основних співвідношень методу скінченних елементів припускають існування лінійної залежності між деформаціями та переміщеннями, тобто використовують лінійний закон такого вигляду:

$$\{\sigma\} = [D](\{\epsilon\} - \{\epsilon_0\}) + \{\sigma_0\}, \quad (2.35)$$

де $\{\sigma\}$ – вектор напружень;

$\{\epsilon\}$ – вектор деформацій;

$\{\sigma_0\}$ – вектор початкових деформацій;

$\{\epsilon_0\}$ – вектор початкових деформацій;

$[D]$ – матриця зв'язків між напруженнями та деформаціями.

Для розв'язання задач про малі деформації використовують й нелінійні рівняння (деякі матеріали мають поведінку, відмінну від закону Гука, вже за малих деформаціях), що можна подати у такому вигляді:

$$F([\sigma], \{\epsilon\}) = 0. \quad (2.36)$$

У випадках, коли співвідношення (2.35) і (2.36) не можливо знайти, використовують ітераційні методи, які часто ґрунтуються на дослідженні з малим кроком по навантаженню.

Метод змінної жорсткості використовують, коли зв'язок між напруженнями і деформаціями, що характеризує поведінку матеріалу, можна подати у вигляді (2.35), але матриця пружності залежить від деформацій, то б то, вона має такий вигляд:

$$[D] = D(\{\epsilon\}) = D(\{d_e\}). \quad (2.37)$$

У такому випадку (2.1) отримає такий вигляд:

$$[K(\{X\})]\{X\} = \{F\}. \quad (2.38)$$

Нехай $\{f\}$ – вектор стовпець, що відповідає малому значенню (кроку по $\{F\}$) навантаження. Для розв'язання (2.38) можна використати таку послідовність кроків, що описують ітераційний процес.

1. Обчислити матрицю жорсткості $[K_0]$ з використанням матриці $[D] = D(\{d_0\})$, де $\{d_0\} = \{0\}$.

2. Поки не досягнуто межі міцності обчислити переміщення $\{d_i\} = [K_{i-1}]^{-1}\{f\}$, матрицю жорсткості $[K_i]$ від $[D] = D(\{d_0\} + \dots + \{d_i\})$.

Потрібно зазначити, що у такому ітераційному процесі можна використовувати стандартні процедуру побудови матриць жорсткості.

Наприклад, якщо вважати зв'язок напруженнями і деформаціями, наприклад, для алюмінієвого сплаву задано у вигляді табл. 2.2, то коефіцієнт пружності для кожного скінченного елемента будемо визначати виходячи з вузлових значень переміщень, обчислених на кожній ітерації за малим навантаженням.

Таблиця 2.2 – Залежність між напруженнями і деформаціями

σ , МПа	0	130	140	150	160	170	180	190	200	210	220	230	270	309	330
ϵ , %	0	0,2	0,24	0,3	0,4	0,55	0,79	1,3	1,5	2,48	3,2	3,61	5,0	10	15

Наприклад, описаний вище ітераційний процес при скінченно-елементному дослідженні паливного бака, який знаходиться під дією внутрішнього тиску, з кроком, рівним 0,01 МПа, дозволяє отримати значення

руйнівного тиску, рівне 0,42 МПа. Для порівняння раніше у лінійній постановці було отримане значення 0,35 МПа. Місце розташування зони ймовірного руйнування баку таке саме (рис. 2.17).

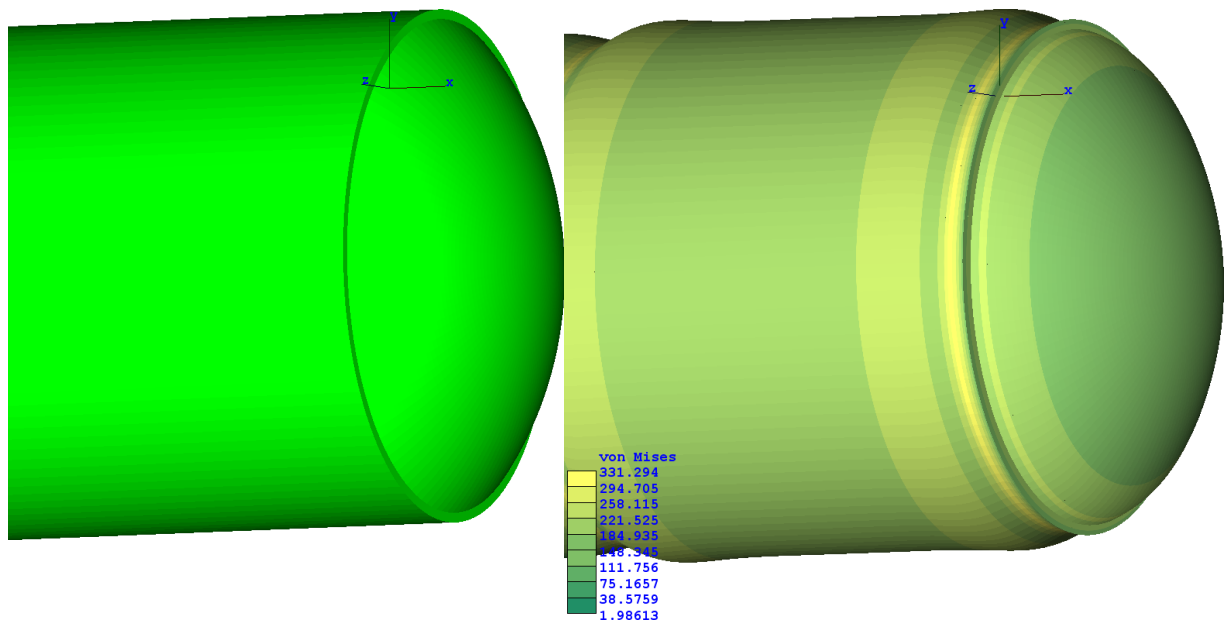


Рисунок 2.17 – Інтенсивність напружень за Мізесом, МПа

2.3. Скінченно-елементне моделювання багатошарових оболонок ракетної техніки із транстропних матеріалів

Оболонки елементів ракетної техніки (наприклад, головні обтикачі, міжступеневі відсіки тощо) виробляють з багатошарових матеріалів. Такі елементи часто виробляють з використанням трьох шарів, серед яких два зовнішні, зазвичай тонкі, виготовлені з більш міцних матеріалів (наприклад, вуглекомпозиту), а внутрішній шар, зазвичай значно товстіший за зовнішні, виготовлений з більш м'якого матеріалу (наприклад, стільниковий папір), який виконує функції термо- і шумоізоляції. Для скінченно-елементного моделювання конструкцій з таких матеріалів нехай справедливі такі гіпотези:

– шари оболонки зроблено із транстропного матеріалу, напружено-деформований стан якого можна визначити шістьма незалежними константами: E_1 – модуль Юнга у першому напрямі, E_2 – модуль Юнга у другому напрямі, ν_{12} – коефіцієнт Пуассона, що відповідає стисненню у другому напрямі при розтягуванні у першому напрямі, G_{12} – модуль зсуву в площині xOy , G_{13} – модуль зсуву в площині xOz , G_{23} – модуль зсуву в площині yOz ;

– шари оболонки ідеально з'єднано один з одним;

- шари оболонки мають постійну товщину і нестисливі;
- переміщення і деформації малі;
- перпендикулярні лінії до серединної поверхні оболонки (нормалі) залишаються прямими після деформації (не обов'язково перпендикулярними);
- нормалі не змінюють власних розмірів після деформації оболонки.

Нехай поле можливих деформацій має вигляд (2.9), а вектор узагальнених переміщень – (2.10). Відповідно, вектори плоских та зсувних деформацій мають вигляд (2.11) та (2.12).

Співвідношення між напруженнями та деформаціями для k -го шару з транстропного матеріалу можна записати у такому вигляді:

$$\{\sigma_k\} = \begin{Bmatrix} \sigma_{x,k} \\ \sigma_{y,k} \\ \tau_{xy,k} \end{Bmatrix} = [D_k] \{\epsilon_k\} = \begin{bmatrix} \frac{E_{1,k}}{1-\nu_{12,k}\nu_{21,k}} & \frac{\nu_{12,k}E_{2,k}}{1-\nu_{12,k}\nu_{21,k}} & 0 \\ \frac{\nu_{21,k}E_{1,k}}{1-\nu_{12,k}\nu_{21,k}} & \frac{E_{2,k}}{1-\nu_{12,k}\nu_{21,k}} & 0 \\ 0 & 0 & G_{12,k} \end{bmatrix} \begin{Bmatrix} \epsilon_{x,k} \\ \epsilon_{y,k} \\ \gamma_{xy,k} \end{Bmatrix}, \quad (2.39)$$

$$\{\tau_k\} = \begin{Bmatrix} \tau_{xz,k} \\ \tau_{yz,k} \end{Bmatrix} = [K_k][G_k]\{\gamma_k\} = \begin{bmatrix} \kappa_1 & 0 \\ 0 & \kappa_2 \end{bmatrix} \begin{bmatrix} G_{13,k} & 0 \\ 0 & G_{23,k} \end{bmatrix} \begin{Bmatrix} \gamma_{xz,k} \\ \gamma_{yz,k} \end{Bmatrix}, \quad (2.40)$$

де $E_{1,k}$, $E_{2,k}$, $\nu_{12,k}$, $G_{12,k}$, $G_{13,k}$ і $G_{23,k}$ – характеристики k -го шару;

$$\nu_{21,k} = \nu_{12,k} \frac{E_{2,k}}{E_{1,k}};$$

κ_1 і κ_2 – множники корекції енергії зсуву у відповідних напрямках.

Потенційну енергію деформації елементарного об'єму можна подати у вигляді (2.15), а узагальнені переміщення у вигляді (2.16)–(2.20). У результаті підстановки (2.16)–(2.20) і (2.39)–(2.40) у (2.15) отримаємо таке співвідношення для потенційної енергії для багатошарової оболонки, що знаходиться під дією тиску q :

$$\begin{aligned} \Pi = & \frac{1}{2} \int_{V_e} ([B_m]\{d_e\} + z[B_b]\{d_e\})^T [D_k] ([B_m]\{d_e\} + z[B_b]\{d_e\}) dV_e + \\ & + \frac{1}{2} \int_{V_e} ([B_c]\{d_e\})^T [K_k][G_k]([B_c]\{d_e\}) dV_e - \int_{S_e} (\{N\}^T \{d_e\}) q dS_e, \end{aligned} \quad (2.41)$$

де матриці зв'язку між узагальненими переміщеннями та деформаціями мають вигляд (2.24)–(2.26).

У результаті мінімізації (2.41) отримаємо такий вираз для потенційної енергії елемента багатошарової оболонки:

$$\int_{V_e} \left([\mathbf{B}_m]^T [\mathbf{D}_k] [\mathbf{B}_m] + z [\mathbf{B}_m]^T [\mathbf{D}_k] [\mathbf{B}_b] + z [\mathbf{B}_b]^T [\mathbf{D}_k] [\mathbf{B}_m] + z^2 [\mathbf{B}_b]^T [\mathbf{D}_k] [\mathbf{B}_b] \right) dV_e + \int_{V_e} [\mathbf{B}_c]^T [\kappa] [\mathbf{G}_k] [\mathbf{B}_c] dV_e = \int_{S_e} \{\mathbf{N}\} q dS_e. \quad (2.42)$$

У результаті інтегрування лівої частини (2.42) за товщиною елемента багатопарової оболонки отримаємо таку формулу для локальної матриці жорсткості скінченного елемента у місцевих координатах:

$$\begin{aligned} [\mathbf{K}_e^{\text{local}}] = & \sum_{k=1}^n \left((z_k - z_{k-1}) \int_{S_e} [\mathbf{B}_m]^T [\mathbf{D}_k] [\mathbf{B}_m] dS_e + \right. \\ & + \frac{1}{2} (z_k^2 - z_{k-1}^2) \int_{S_e} ([\mathbf{B}_m]^T [\mathbf{D}_k] [\mathbf{B}_b] + [\mathbf{B}_b]^T [\mathbf{D}_k] [\mathbf{B}_m]) dS_e + \\ & \left. + \frac{1}{3} (z_k^3 - z_{k-1}^3) \int_{S_e} [\mathbf{B}_b]^T [\mathbf{D}_k] [\mathbf{B}_b] dS_e + (z_k - z_{k-1}) [\kappa] \int_{S_e} [\mathbf{B}_c]^T [\mathbf{G}_k] [\mathbf{B}_c] dS_e \right). \end{aligned} \quad (2.43)$$

Перехід до локальної матриці жорсткості у глобальних координатах можна виконати з використанням формул (2.27)–(2.29). Також справедливим залишається зауваження про використання фіктивної жорсткості для позицій θ_z локальної матриці жорсткості у місцевих координатах.

2.4. Приклад дослідження напружено-деформованого стану трипарової оболонки ракети-носія, яка знаходиться під дією комбінованого навантаження

Для оболонкової конструкції, яка знаходиться під дією зовнішнього тиску q , вектор-стовпець можна обчислити за допомогою формул (2.33) і (2.34). Якщо оболонка знаходиться під дією осьової сили, що стискає, або сили $\vec{f}$, що перерізує, яка зосереджено діє у центрі конструкції (у перерізі). Водночас навантаження на поверхню оболонки передають елементи зв'язку типу RBE3 (моделювання дії корисної ваги, яку закріплено гумометалевими з'єднаннями), то вузлові значення сили у відповідних вузлах дискретної моделі можна обчислити з використанням такої формули:

$$\{\mathbf{F}\} = \begin{cases} \frac{\vec{f}}{n}, & \text{якщо вузел з'єднано з центром дії сили,} \\ 0, & \text{інакше,} \end{cases}$$

де n – кількість вузлів, з'єднаних з точкою прикладення сили.

Якщо аналогічним чином у центрі конструкції діє момент $\vec{m}$, що згинає, то вузлові значення сили у відповідних вузлах дискретної моделі можна обчислити з використанням такої формули:

$$\{\mathbf{F}\} = \begin{cases} \frac{\vec{m} \times \mathbf{r}_i}{\sum_i |\mathbf{r}_i|^2}, & \text{якщо вузел } i \text{ з'єднано з центром дії моменту,} \\ 0, & \text{інакше,} \end{cases}$$

де $\mathbf{r}$ – радіус-вектор, що з'єднує точку прикладення моменту з залежним вузлом на поверхні оболонки.

Як приклад розглянемо задачу дослідження поведінки головного обтикача ракети. Головний обтикач можна подати двома оболонками: конічною та циліндричною, з'єднаних в основах однакового радіуса. На конструкцію діють одночасно момент M_p , поперечна сила Q_p та осьова сила, що стискає, T_p . Навантаження діють зосереджено у центрах перерізів згідно зі схемою, наведеною на рис. 2.18, а. Сила Q_p і момент M_p діють в одному напрямі (рис. 2.18, б). Навантаження на поверхню оболонки діє через гнучке оздоблення (елементи зв'язку типу RBE3).

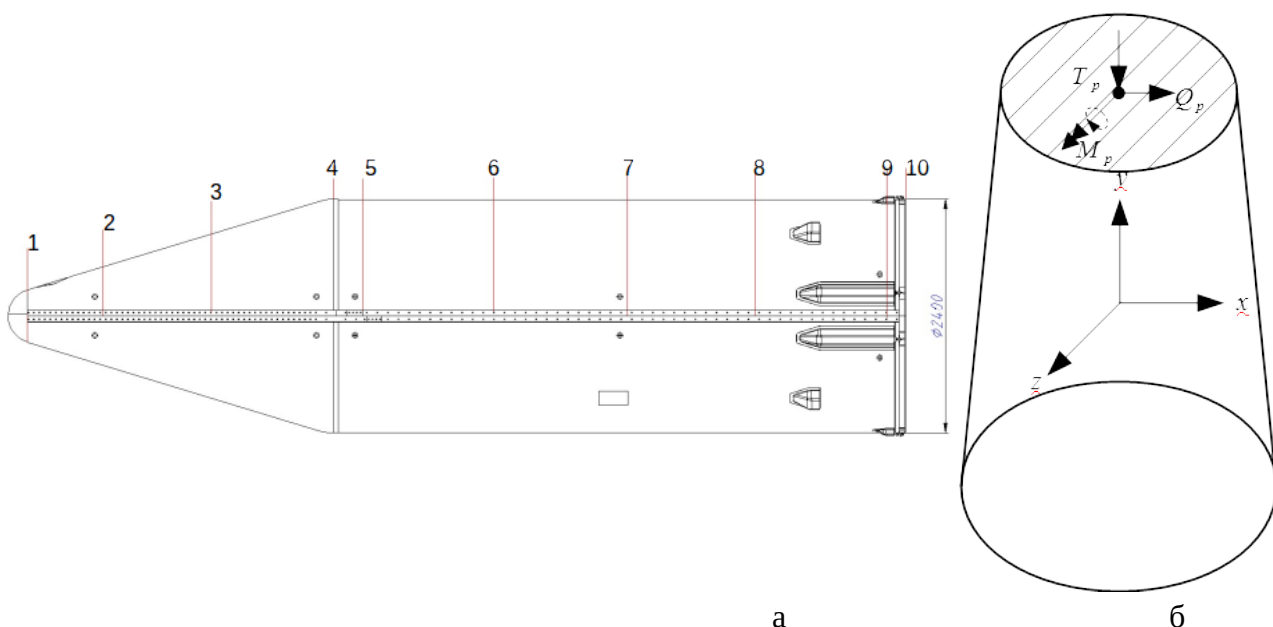


Рисунок 2.18 – Схема дії навантажень:
а – перерізи дії навантаження; б – вектори сил

Нехай на сили діють на головний обтикач відповідно до закону, що поданий у табл. 2.3.

Таблиця 2.3 – Навантаження на обтикач

Переріз	d , м	M_p , тс·м	Q_p , тс	T_p , тс
1	0,536	0,02	0,21	0,48
2	0,986	0,51	1,13	1,17
3	1,678	3,11	3,38	2,60
4	2,329	8,83	6,49	2,59
5	2,400	11,35	7,56	2,32
6	2,400	22,89	8,29	2,78
7	2,400	34,28	8,30	3,27
8	2,400	45,60	8,48	3,37
9	2,400	57,63	9,45	3,66
10	2,400	59,52	9,51	3,93

Розрахунок виконано із припущення, що головний обтикач є тришаровою конструкцією. Зовнішні шари мають товщину 0,00114 м і такі характеристики: $E_1=E_2=34000$ МПа, $G_{12}=3500$ МПа, $G_{13}=G_{23}=800$ МПа. Внутрішній шар має товщину 0,0254 м і такі характеристики: $E_1=E_2=128$ МПа і $G_{12}=G_{13}=G_{23}=117$ МПа. Результати скінченно-елементного моделювання зведено у табл. 2.4.

Таблиця 2.4 – Напружено-деформований стан обтикача

	Основу жорстко закріплено		Основу шарнірно закріплено	
	min	max	min	max
u , м	0,000	0,096	0,000	0,097
v , м	-0,013	0,015	-0,013	0,015
w , м	-0,001	0,001	-0,001	0,001
θ_x , радіан	-0,004	0,011	-0,009	0,004
θ_y , радіан	-0,014	0,014	-0,014	0,014
θ_z , радіан	-0,005	0,005	-0,004	0,004
σ_x , МПа	-17,872	15,420	-8,115	7,180
σ_y , МПа	-202,436	185,372	-177,040	161,170
σ_z , МПа	-44,517	40,886	-15,415	14,035
τ_{xy} , МПа	-38,799	6,616	-39,474	6,662
τ_{xz} , МПа	-17,397	17,397	-7,461	7,461

	Основу жорстко закріплено		Основу шарнірно закріплено	
	min	max	min	max
τ_{yz} , МПа	-19,354	19,354	-19,630	19,630
σ_i , МПа	0,347	185,670	0,349	179,704

На рис. 2.19 порівняно картини напружено-деформованого стану конструкції за різних умов закріплення.

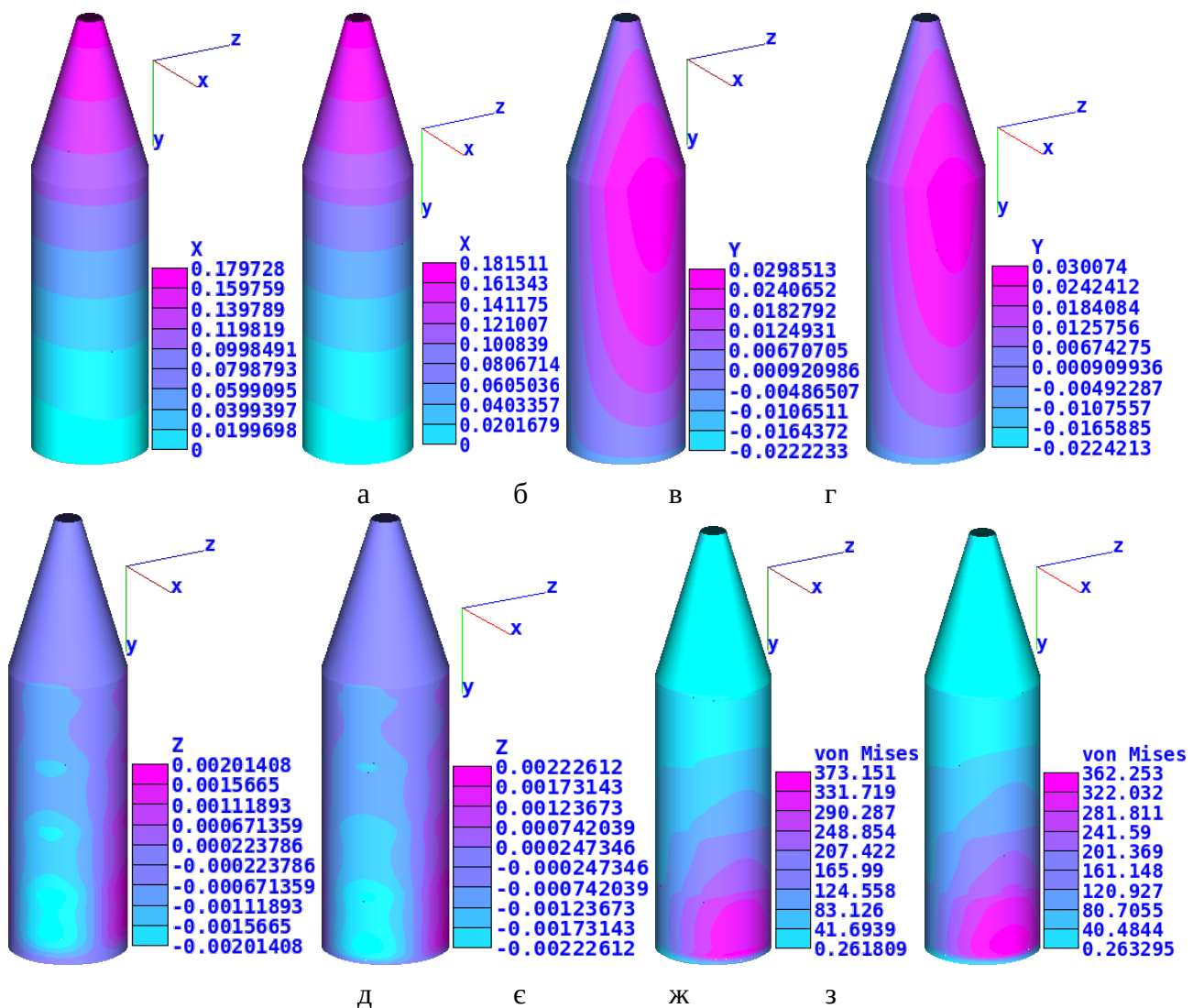


Рисунок 2.19 – Напружено-деформований стан обтикача:

- а – u , основу жорстко закріплено; б – u , основу шарнірно закріплено;
- в – v , основу жорстко закріплено; г – v , основу шарнірно закріплено;
- д – w , основу жорстко закріплено; е – w , основу шарнірно закріплено;
- ж – σ_i , основу жорстко закріплено; з – σ_i , основу шарнірно закріплено

2.5. Аналітико-чисельні методи у хмарних САПР

При розрахунках напружено-деформованого стану конструкцій у хмарних САПР важливою задачею є верифікація отриманих результатів. Одним з підходів тут може бути порівняння з деякими іншими моделями, які розроблені з використанням альтернативних підходів. Для цього можуть використовуватися модельні задачі, розв'язок яких попередньо отримано аналітичними або аналітико-чисельними методами. Наприклад, згин одношарових та тришарових круглих пластин, балок, циліндричних оболонок.

Для задач фізично нелінійного деформування, стійкості та динаміки елементів конструкцій традиційно використовуються асимптотичні методи, наприклад, метод малого параметра, у комбінації з методами Бубнова-Гальоркіна, ВКБ-Гальоркіна.

Отже, поєднання перерахованих наближених аналітичних підходів з методом скінченних елементів у модулях хмарних САПР забезпечує можливості оцінки діапазону значень розв'язку. При одночасному використанні аналітичних і чисельних методів у хмарі користувач отримає більше інформації щодо характеристик об'єкта, що проектується.

2.5.1 Наближене аналітичне розв'язання задачі стійкості тришарової конічної оболонки при комбінованому навантаженні

До сучасних інженерних конструкцій пред'являється вимога високої міцності і жорсткості при мінімальній матеріаломісткості. Такі властивості мають багатошарові конструкції, серед яких особливе місце займають тришарові оболонки, що складаються з двох тонкостінних досить жорстких несучих шарів і шару-заповнювача малої щільності та жорсткості.

Істотний внесок в розв'язання задач стійкості однорідних і багатошарових оболонок вкладений значною кількістю вітчизняних і зарубіжних дослідників. Дослідження в галузі стійкості тришарових оболонок виконано у роботах Г. В. Закирова [315], С. О. Амбарцумяна [316], Е. І. Григолюка, П. П. Чулкова [317], Я. М. Григоренко [318], О. М. Андрєєва [319], С. М. Сухиніна [320], E. Tertel, P. Kuryło [321], M. A. Boorboor, Ajdari, S. Jalili, M. Jafari, J. Zamani і M. Shariyat [322] та інші, пов'язані з предметом даного дослідження.

Слід зазначити, що проблема стійкості тришарових конічних оболонок при сумісній дії трьох навантажень, навіть в лінійній постановці, досліджена не повною мірою і представляє науковий і практичний інтерес, особливо в

питаннях побудови наближених аналітичних розв'язків і апроксимації граничної поверхні, що відділяє область стійкості від області нестійкості досліджуваної системи.

Істотним моментом дослідження є зведення повної системи рівнянь стійкості тришарової оболонки, наведеної в монографії Е. І. Григолюка, П. П. Чулкова [317, с. 142] до розв'язувального сингулярного звичайного диференціального рівняння шостого порядку відносно функції переміщень зі змінними коефіцієнтами. Слід зазначити, що аналоги знаходження розв'язку таких рівнянь, відомі лише для одношарових оболонок [323, 324].

Для побудови ефективного наближеного аналітичного розв'язку задачі, використовуються три підходи. Перший ґрунтується на асимптотичному підході на базі методів фазових інтегралів (у літературі іменований наближенням Ліувіля-Гріна, методом ВКБ (Вентцеля-Крамерса-Бриллюэна та ін.) [325], ефективність застосування якого для розрахунку стійкості конічних оболонок з різними структурними характеристиками продемонстрована в монографії [324]. Другий підхід включає гібридний ВКБ-Гальоркін метод, ідея якого бере початок з робіт J. F. Geer, C. M. Andersen [326], що базуються на поєднанні методу збурень і принципу ортогоналізації за Гальоркіним. Потім гібридний метод вдосконалений в роботах [317, 318, 327, 328] для розв'язання прикладних задач математичної фізики, що зводяться до сингулярних лінійних і, у поєднанні з методом збурень, нелінійних диференціальних рівнянь зі змінними коефіцієнтами, у тому числі, до розв'язання задачі стійкості однорідної конічної оболонки [328, 329]. Третій підхід реалізує чисельний метод кінцевих різниць. Порівняння отриманих розв'язків, відповідно до вказаних методів, дозволило виявити найбільш ефективний аналітичний метод дослідження.

2.5.1.1 Постановка задачі

Розглядається стійкість тонкої пружної тришарової зрізаної конічної оболонки кругового перерізу при спільній дії рівномірного зовнішнього тиску, осьового стискання і кручення (рис. 2.20). Вивчається оболонка, стінки якої складаються з ізотропних несучих шарів і трансверсально ізотропного заповнювача. У крайових умовах передбачається, що торці оболонки вільно оперті, а також є діафрагма нескінченної жорсткості, що перешкоджає відносному зсуву несучих шарів уздовж краю оболонки.

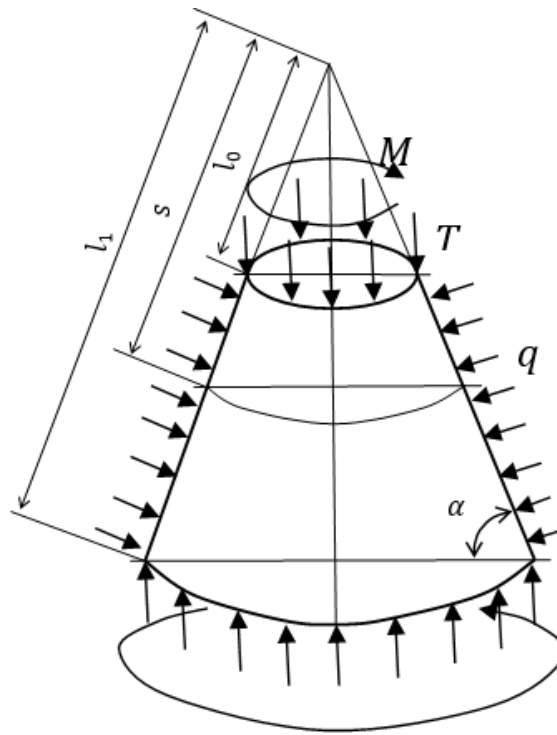


Рисунок 2.20 – Схема навантаження конічної оболонки

Наслідуючи роботу [317], для несучих шарів використовується гіпотеза Кірхгофа-Лява прямої нормалі, а для заповнювача – гіпотеза про нестискуваність матеріалу в поперечному напрямі, при додатковому припущенні, що деформація поперечного зсуву по товщині заповнювача розподілена за деяким відомим законом $\varphi(z)$. Для тришарової оболонки прийняті зведені коефіцієнт Пуассона і модуль пружності.

2.5.1.2 Фізичні характеристики тришарової оболонки

Несучий шар оболонки, розташований з боку зовнішньої нормалі, іменується першим шаром, шар з боку внутрішньої нормалі – другим, а заповнювач – третім. Вводяться позначення [317, с. 50]: h – товщина стінки оболонки; h_j – товщина шарів (рис. 2.21); $t_j = h_j h^{-1}$ – безрозмірна товщина шарів; E_j і ν_j – модулі пружності і коефіцієнти Пуассона шарів;

$\rho_j = \frac{E_j h_j}{1 - \nu_j^2} \left(\sum_{j=1}^3 \frac{e_j h_j}{1 - \nu_j^2} \right)^{-1}$ ($j = \{1, 2, 3\}$) – безрозмірні характеристики жорсткості;

$\nu = \sum_{j=1}^3 \nu_j \rho_j$, $E = (1 - \nu^2) h^{-1} \left(\sum_{j=1}^3 \frac{e_j h_j}{1 - \nu_j^2} \right)^{-1}$ – зведені коефіцієнт Пуассона і модуль

пружності оболонки; G - модуль зсуву оболонки;

$$\varphi_1'(z)=1 \text{ і } \varphi_2'(z)=1-\frac{z^2}{c^2}. \quad (2.44)$$

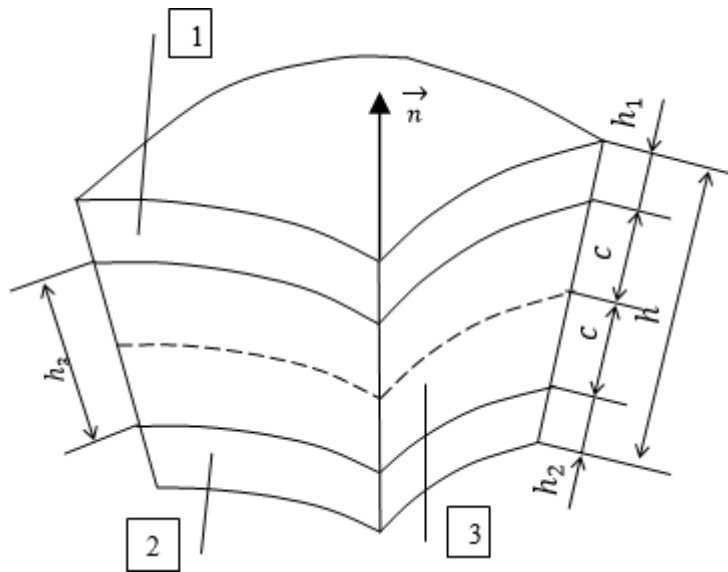


Рисунок 2.21 – Схема тришарової поверхні

Використовується перший закон розподілу $\varphi(z)$, який відповідає випадку, коли нормаль до початкової поверхні у заповнювачі в процесі деформації оболонки повертається, але не викривляється, не деформується в поперечному напрямі. При цьому, основні параметри, що відповідають першому закону (2.44), визначаються формулами [317, с.67]:

$$\begin{aligned} \theta_1 &= \Theta_1, \\ \theta_2 &= \Theta_1 + \Theta_2, \\ \theta_3 &= \Theta_1 + 2\Theta_2 + \Theta_3 = \Theta, \\ v &= \frac{\theta_1 \theta_3 - \theta_2^2}{\theta_1 \theta_3}, \end{aligned} \quad (2.45)$$

а циліндрична жорсткість складеної оболонки – формулою

$$D = \frac{E h^3}{12(1-\nu^2)} \Theta, \quad (2.46)$$

$$\begin{aligned} \text{де } \Theta_1 &= t_3^2 (1 + 2(\rho_1 + \rho_2) - 3(\rho_1 - \rho_2)^2); \\ \Theta_2 &= 3t_3 \rho_3 (t_1 \rho_1 + t_2 \rho_2) + 6\rho_1 \rho_2 t_3 (t_1 + t_2); \\ \Theta_3 &= 4(\rho_1 t_1^2 + \rho_2 t_2^2) - 3t_3^2 (\rho_1 t_1 - \rho_2 t_2)^2. \end{aligned}$$

Вводиться параметр зсуву, який для першого закону (2.44) має вигляд [317, с. 69]:

$$\beta = \frac{12 G t_3 (1 - \nu^2)}{E \Theta_1}. \quad (2.47)$$

2.5.1.3 Розв'язувальне диференціальне рівняння

Систему рівнянь стійкості тришарової конічної оболонки беремо у формі [317, с. 142]:

$$\begin{cases} D \left(1 - \frac{\nu h^2}{\beta} \Delta \right) \Delta^2 \psi + \frac{\tan \alpha}{s} \frac{\partial^2}{\partial s^2} F = g, \\ \Delta^4 F = E h \frac{\tan \alpha}{s} \frac{\partial^2}{\partial s^2} \left(1 - \frac{h^2}{\beta} \Delta \right) \psi, \end{cases} \quad (2.48)$$

де α – кут за основою конуса;

s – відстань уздовж твірної конуса від вершини до точки серединної поверхні;

l_0 і l_1 – відстані до меншої та більшої основ відповідно (рис. 2.20);

$x = \frac{s}{l_1}$, φ – полярний кут;

n – число хвиль в окружному напрямі при витріщенні оболонки;

w – нормальне переміщення точок серединної поверхні;

F – функція зусиль;

q – зовнішній нормальний тиск, що діє як на бічну поверхню, так і на основи оболонки;

T – осьова сила, перпендикулярна основам;

M – крутильний момент;

$w = \left(1 - \frac{h^2}{\beta} \Delta \right) \psi$ – залежність між функціями w і ψ ;

g – фіктивне поперечне навантаження, яке дорівнює:

$$g = -(N_1 \kappa_1 + N_2 \kappa_2 + 2 T_{12} \kappa_{12}); \quad (2.49)$$

N_1 , N_2 , T_{12} – погонні нормальні і дотичні зусилля в основному стані:

$$N_1 = \frac{q}{2 \tan \alpha} \left(\frac{l_0^2}{l_1 x} - x l_1 \right) - \frac{T}{\pi l_1 x \sin(2\alpha)}, \quad N_2 = \frac{-q l_1 x}{\tan \alpha}, \quad T_{12} = \frac{M}{\pi x^2 l_1^2 \cos^2 \alpha}; \quad (2.50)$$

$$\begin{aligned}\kappa_1 &= -\frac{1}{l_1^2} \frac{\partial^2 w}{\partial x^2}, \\ \kappa_2 &= -\frac{1}{l_1^2} \left(\frac{1}{x^2 \cos^2 \alpha} \frac{\partial^2 \psi}{\partial \varphi^2} + \frac{1}{x} \frac{\partial w}{\partial x} \right), \\ \kappa_{12} &= -\frac{1}{l_1^2 x \cos \alpha} \left(\frac{\partial^2 w}{\partial x \partial \varphi} - \frac{1}{x} \frac{\partial w}{\partial x} \right);\end{aligned}\quad (2.51)$$

Δ – оператор, що визначається формулою:

$$\Delta = \frac{1}{l_1^2} \left(\frac{1}{x} \frac{\partial}{\partial x} + \frac{1}{x^2 \cos^2 \alpha} \frac{\partial^2}{\partial \varphi^2} + \frac{\partial^2}{\partial x^2} \right). \quad (2.52)$$

Наслідуючи роботи [323, с. 629], [324, с. 55], для отримання розв'язку системи (2.48) функції ψ і F подаються у вигляді:

$$\psi(x, \varphi) = \Psi(x) \tan(\alpha) \cos(\gamma(1-x) + n\varphi), \quad (2.53)$$

$$F(x, \varphi) = \Phi(x) \varepsilon^2 l_1 E h \tan^2(\alpha) \cos(\gamma(1-x) + n\varphi), \quad (2.54)$$

де

$$\varepsilon = \sqrt{\frac{h \cot \alpha}{l_1 \sqrt{12(1-\nu^2)}}}, \quad \gamma = \frac{n}{\cos \alpha} \gamma_1, \quad (2.55)$$

γ_1 – тангенс кута нахилу гребеня хвилі до твірної.

Передбачається, що в комбінації зовнішніх навантажень переважаючу дію має тиск. В цьому випадку хвилеутворення вздовж твірної оболонки виражене набагато слабкіше, ніж в окружному напрямі. Водночас для оболонок середньої довжини число хвиль в окружному напрямі таке, що $n^2 \gg 1$, а в подовжньому напрямі утворюється одна півхвиля.

Аналогічно [323] і [324] вводяться безрозмірні параметри $\delta = n^2 / \cos^2 \alpha$, $p = \varepsilon \delta$ і безрозмірні зусилля

$$\chi = \frac{q l_1}{E h \varepsilon^3 \tan^3 \alpha}, \quad \eta = \frac{T \cos \alpha}{2 \pi E h \varepsilon^2 l_1 \sin^3 \alpha}, \quad \lambda = \frac{M}{2 \pi l_1^2 E h \varepsilon^{5/2} \sin^2 \alpha}, \quad (2.56)$$

а також, аналогічно [315], параметр зсуву має такий вигляд:

$$K = \frac{h^2}{\beta l_1^2}. \quad (2.57)$$

Система рівнянь (2.48) з урахуванням співвідношень (2.46), (2.49)–(2.57) інтегрується по координаті φ методом Бубнова-Гальоркіна, тобто обидві частини кожного з рівнянь системи множаться на $\cos(\gamma(1-x)+n\varphi)$ і інтегруються від 0 до 2π , що дозволяє позбавитися від змінної φ . Після перетворень, враховуючи, що $\varepsilon \ll 1$, можна прийти до системи, в якій виключені доданки, що містять ε у степенях, вищих за першу:

$$\left(\begin{aligned} & -\frac{\varepsilon \gamma^2}{K} \Phi + \frac{\varepsilon}{K} \Phi'' + \left[\frac{p^3 v \Theta}{x^5} + \left(-\frac{20v}{x^2} + \frac{1}{K} + 3v\gamma^2 \right) \frac{\Theta \varepsilon p^2}{x^3} - \frac{\chi p^2}{x^2} - \frac{\lambda \gamma p^{3/2}}{x^4} + \right. \\ & + \left(-\gamma^2 + \frac{6}{x^2} \right) \frac{\eta p}{x^2} + \left(-\frac{3\gamma^2}{2} - \frac{\gamma^2 l_0^2}{2l_1^2 x^2} + \frac{1}{x^2} - \frac{1}{K} - \frac{3l_0^2}{l_1^2 x^4} \right) \chi \varepsilon p + \\ & + \left(-\frac{2}{x^2} - 1 - \gamma^2 \right) \frac{\gamma \lambda p^{1/2}}{x^2} + \left(-\frac{2}{x^2} - \frac{1}{K} - \gamma^2 \right) \gamma^2 \eta \varepsilon \left. \right] \Psi + \\ & + \left[\frac{9\varepsilon p^2 v \Theta}{x^4} + \left(\frac{2\varepsilon \chi l_0^2}{l_1^2} - 4\eta \right) \frac{p}{x^3} + \left(3\gamma^2 - \frac{2}{x^2} \right) \frac{\eta \varepsilon}{x} \right] \Psi' + \\ & + \left[-\frac{3p^2 v \Theta}{x^3} + \left(\frac{3}{2} - \frac{l_0^2}{2l_1^2 x^2} \right) \chi p + n p + \frac{3\lambda p^{1/2}}{x^2} + \left(6\gamma^2 + \frac{2}{x^2} + \frac{1}{K} \right) \eta \right] \varepsilon \Psi'' - \\ & - \frac{\eta \varepsilon}{x} \Psi''' - \eta \varepsilon \Psi^{IV} = 0; \\ & \frac{\varepsilon p^2}{x^3} \Phi + \left[\left(\frac{\gamma^2}{x^2} - \frac{6}{x^4} \right) p + \left(\gamma^4 + \frac{\gamma^2}{K} + \frac{2\gamma^2}{x^2} \right) \varepsilon \right] \Psi + \left[\frac{4p}{x^3} + \left(\frac{2}{x^3} - \frac{3\gamma^2}{x} \right) \varepsilon \right] \Psi' + \\ & + \left[-\frac{p}{x^2} + \left(-\frac{1}{K} - 6\gamma^2 - \frac{2}{x^2} \right) \varepsilon \right] \Psi'' + \frac{\varepsilon}{x} \Psi''' + \varepsilon \Psi^{IV} = 0. \end{aligned} \right) \quad (2.58)$$

Виключення з системи (2.58) функції $\Phi(x)$ дозволяє отримати розв'язувальне звичайне диференціальне рівняння шостого порядку відносно функції переміщення $\Psi(x)$ зі змінними коефіцієнтами:

$$\begin{aligned} & Q_0(x) \Psi + Q_1(x) \Psi' + Q_2(x) \Psi'' + \\ & + Q_3(x) \Psi''' + Q_4(x) \Psi^{IV} + Q_5(x) \Psi^V + Q_6(x) \Psi^VI = 0, \end{aligned} \quad (2.59)$$

де

$$\begin{aligned}
Q_0(x) &= \frac{p^5 \nu \Theta}{x^5} + \left(-\frac{20\nu}{x^2} + \frac{1}{K} + 3\nu y^2 \right) \frac{\Theta \varepsilon p^4}{x^3} - \frac{\chi p^4}{x^2} - \frac{\lambda y p^{7/2}}{x^4} + \\
&+ \left(-y^2 + \frac{6}{x^2} \right) \frac{\eta p^3}{x^2} + \left(-\frac{3y^2}{2} - \frac{y^2 l_0^2}{2l_1^2 x^2} + \frac{1}{x^2} - \frac{1}{K} - \frac{3l_0^2}{l_1^2 x^4} \right) \chi \varepsilon p^3 + \\
&+ \left(-\frac{2}{x^2} - 1 - y^2 \right) \frac{y \lambda p^{5/2}}{x^2} + \left(-\frac{2}{x^2} - \frac{1}{K} - y^2 \right) y^2 \eta \varepsilon p^2 + \\
&+ \left(-\frac{6y^2}{x} + \frac{12}{x^3} + x y^4 \right) p + \left(x^3 y^6 - 4x y^4 + \frac{x^3 y^4}{K} - \frac{6x y^2}{K} \right) \varepsilon, \\
Q_1(x) &= \frac{9 \varepsilon p^4 \nu \Theta}{x^4} + \left(\frac{2 \varepsilon \chi l_0^2}{l_1^2} - 4 \eta \right) \frac{p^3}{x^3} + \left(3y^2 - \frac{2}{x^2} \right) \frac{\eta \varepsilon p^2}{x} + \left(2y^2 - \frac{12}{x^2} \right) p + \\
&+ \left(4y^2 - 9x^2 y^4 - \frac{6x^2 y^2}{K} \right) \varepsilon, \\
Q_2(x) &= -\frac{3p^4 \nu \Theta \varepsilon}{x^3} + \left(\frac{3}{2} - \frac{l_0^2}{2l_1^2 x^2} \right) \chi \varepsilon p^3 + \eta \varepsilon p^3 + \frac{3 \varepsilon \lambda y p^{5/2}}{x^2} + \left(\frac{6}{x} - 2x y^2 \right) p + \\
&+ \left(6y^2 + \frac{2}{x^2} - \frac{1}{K} \right) \varepsilon \eta p^2 + \left(-7x^3 y^4 - \frac{2x^3 y^2}{K} + 44x y^2 + \frac{6x}{K} \right) \varepsilon, \\
Q_3(x) &= -\frac{\eta \varepsilon p^2}{x} - 2p + \left(40y^2 x^2 + \frac{6x^2}{K} \right) \varepsilon, \\
Q_4(x) &= -\eta \varepsilon p^2 + p x + \left(-8x + 7x^3 y^2 + \frac{x^3}{K} \right) \varepsilon, \\
Q_5(x) &= -7x^2 \varepsilon, \\
Q_6(x) &= -x^3 \varepsilon.
\end{aligned}$$

У крайових умовах передбачається, що торці вільно оперті, а також є діафрагма нескінченної жорсткості, що перешкоджає відносному зсуву несучих шарів уздовж краю оболонки:

$$\Psi = \frac{\partial^2 \Psi}{\partial x^2} = \frac{\partial^4 \Psi}{\partial x^4} = 0, \text{ якщо } x = \frac{l_0}{l_1} \text{ і } x = 1. \quad (2.60)$$

2.5.1.4 Застосування ВКБ методу для розв'язання рівняння

Розв'язання ВКБ методом будується відповідно до [324, 325]. Оскільки $n^2 \gg 1$, то δ є великим параметром. Зазначений метод припускає подання розв'язку рівняння (2.59) у вигляді:

$$\Psi(x) = \exp\left(\int_{l_0/l_1}^x \varphi(t) dt\right), \quad (2.61)$$

$$\varphi(x) = \sum_{k=0}^{\infty} \varphi_k(x) \delta^{1-k}. \quad (2.62)$$

Після підстановки (2.61) і (2.62) у рівняння (2.59) і ділення отриманого рівняння на функцію $\Psi(x)$, одержимо:

$$\begin{aligned} & \delta^6 \left[-x^3 \varepsilon (\varphi_0(x))^6 - \eta \varepsilon^3 (\varphi_0(x))^4 - \frac{3\varepsilon^5 \nu \Theta}{x^2} (\varphi_0(x))^2 \right] + Q_0(x) + \\ & + \delta^5 f_5(x, \varphi_0, \varphi_1) + \delta^4 f_4(x, \varphi_0, \varphi_1, \varphi_2) + \dots + \delta f_1(x, \varphi_0, \varphi_1, \dots, \varphi_5) + \\ & + f_1(x, \varphi_0, \varphi_1, \dots, \varphi_6) + \dots + \delta^{-1} f_{-1}(x, \varphi_0, \varphi_1, \dots, \varphi_7) + \dots + \\ & + \delta^{4.5} f_{4.5}(x, \varphi_0) + \delta^{3.5} f_{3.5}(x, \varphi_0, \varphi_1) + \dots + \delta^{0.5} f_{0.5}(x, \varphi_0, \varphi_1, \dots, \varphi_4) + \\ & + \delta^{-0.5} f_{-0.5}(x, \varphi_0, \varphi_1, \dots, \varphi_5) + \dots = 0. \end{aligned}$$

У нульовому наближенні маємо алгебраїчне рівняння шостого степеня відносно функції $\varphi_0(x)$:

$$\delta^6 \left[-x^3 \varepsilon (\varphi_0(x))^6 - \eta \varepsilon^3 (\varphi_0(x))^4 - \frac{3\varepsilon^5 \nu \Theta}{x^2} (\varphi_0(x))^2 \right] + Q_0(x) = 0. \quad (2.63)$$

Після заміни $\bar{\varphi}(x) = (\varphi_0(x))^2$ порядок рівняння (2.63) знижується до третього. У даному випадку рівняння третього степеня з від'ємним дискримінантом на відрізку $[l_0/l_1, 1]$ має один дійсний і два комплексно спряжених корені. Функція, що є дійснозначним його коренем, змінює знак при переході через деяку точку $x \in [l_0/l_1, 1]$. Отже, зворотна заміна призводить до коренів рівняння (2.63):

$$\varphi_{0,k_1}(x) = \begin{cases} \pm F_1(x), & x \in [x_0, x_1], \\ \pm i F_1(x), & x \in [x_1, 1], \end{cases} \quad (k_1 = 1, 2);$$

$$\varphi_{0,k_2}(x) = \pm F_2(x) \pm i F_3(x), (k_2 = \overline{3, 6}). \quad (2.64)$$

Якщо в розкладі (2.62) утримувати тільки нульове наближення, то асимптотичним ВКБ-розв'язком рівняння (2.59) буде функція

$$\begin{aligned} \Psi(x) &= \sum_{k=1}^6 C_k \omega_k(x); \\ \omega_1(x) &= \begin{cases} \sinh S_1(x), & x < x_1, \\ \sin S_1(x), & x \geq x_1, \end{cases} \\ \omega_2(x) &= \begin{cases} \cosh S_1(x), & x < x_1, \\ \cos S_1(x), & x \geq x_1, \end{cases} \\ \omega_3(x) &= e^{S_2(x)} \sin S_3(x), \\ \omega_4(x) &= e^{S_2(x)} \cos S_3(x), \\ \omega_5(x) &= e^{-S_2(x)} \sin S_3(x), \\ \omega_6(x) &= e^{-S_2(x)} \cos S_3(x), \end{aligned} \quad (2.65)$$

де

$$\begin{aligned} S_1(x) &= \delta \int_{x_1}^x F_1(t) dt, \\ S_k(x) &= \delta \int_{l_0/l_1}^x F_k(t) dt, \quad (k=2, 3). \end{aligned}$$

Підстановка у граничні умови (2.60) функції (2.65) призводить до системи шести лінійних алгебраїчних рівнянь відносно коефіцієнтів $C_k, k = \overline{1, 6}$. Прирівнювання її визначника до нуля, дає можливість отримати характеристичне рівняння для визначення критичних комбінацій навантажень.

2.5.1.5 Застосування гібридного ВКБ-Гальоркін методу для розв'язання рівняння

Перший крок гібридного асимптотичного підходу припускає використання методу ВКБ, результатом якого є відшукування скінченного числа функцій $\varphi_k(x)$ розкладу (2.62). На другому кроці ці функції використовують як координатні для методу Гальоркіна. У даному випадку як координатна функція

$\varphi(x)$ розглядається функція $F_1(x)$ з рівності (2.64). Розв'язок рівняння (2.59) подається у вигляді:

$$\Psi(x) = \exp\left(\int_{l_0/l_1}^x \Lambda(\delta) \varphi_0(t) dt\right), \quad (2.66)$$

де $\Lambda(\delta)$ – невідома функція параметра δ .

Підстановка функції $\Psi(x)$, визначеної рівністю (2.66), в рівняння (2.59) дозволяє звести ліву частину цього рівняння до вигляду

$$\exp\left(\int_{l_0/l_1}^x \Lambda(\delta) \varphi_0(t) dt\right) R(x, \varphi_0, \varphi_0', \dots, \varphi_0^{(5)}, \Lambda, \delta),$$

де

$$R(x, \varphi_0, \varphi_0', \dots, \varphi_0^{(5)}, \Lambda, \delta) = \sum_{k=1}^6 \Lambda^k Y_k(x, \varphi_0, \dots, \varphi_0^{(6-k)}) + Y_0(x), \quad (2.67)$$

$$Y_0(x) = Q_0(x).$$

У загальному випадку функція (2.66) не задовольняє рівняння (2.59), а функція R вигляду (2.67) не перетворюється на нуль. Відповідно до методу Гальоркіна, значення $\Lambda(\delta)$ знаходиться з умови ортогональності функцій R і φ_0 , тобто

$$\int_{l_0/l_1}^1 \varphi_0(x) R(x, \varphi_0, \varphi_0', \dots, \varphi_0^{(5)}, \Lambda, \delta) dx = 0. \quad (2.68)$$

Рівняння (2.68) набуває вигляду:

$$\Lambda^6 a_6 + \dots + \Lambda a_1 + a_0 = 0, \quad (2.69)$$

$$a_s = \int_{l_0/l_1}^1 \varphi_0(x) Y_s dx, \quad (s = \overline{0, 6}).$$

Алгебраїчне рівняння (2.69) має два дійсні корені: Λ_1 і Λ_2 , та дві пари комплексно-спряжених: $\Lambda_{3,4} = \xi_1 \pm i\zeta_1$ і $\Lambda_{5,6} = \xi_2 \pm i\zeta_2$.

Отже, розв'язок гібридного ВКБ-Гальоркін методу має вигляд:

$$\Psi(x) = C_1 e^{\Lambda_1 \psi(x)} + C_2 e^{\Lambda_2 \psi(x)} + e^{\xi_1 \psi(x)} (C_3 \sin \varsigma_1 \psi(x) + C_4 \cos \varsigma_1 \psi(x)) + e^{\xi_2 \psi(x)} (C_5 \sin \varsigma_2 \psi(x) + C_6 \cos \varsigma_2 \psi(x)), \quad (2.70)$$

$$\text{де } \psi(x) = \int_{l_0/l_1}^x \varphi_0(t) dt.$$

Урахування крайових умов (2.60) для функції $\Psi(x)$ вигляду (2.70) дозволяє отримати характеристичне рівняння.

2.5.1.6 Застосування методу скінченних різниць для розв'язання рівняння

Метод скінченних різниць чисельного розв'язання рівняння (2.59) використовується для порівняння результатів, отриманих аналітичними методами з чисельним. Для звичайного диференціального рівняння шостого порядку МКР припускає використання центральних скінченних різниць третього порядку у внутрішніх точках розбиття відрізка $[l_0/l_1; 1]$:

$$\begin{aligned} \Psi'(x_k) &\approx \frac{1}{60H} (\Psi_{k+3} - 9\Psi_{k+2} + 45\Psi_{k+1} - 45\Psi_{k-1} + 9\Psi_{k-2} - \Psi_{k-3}), \\ \Psi''(x_k) &\approx \frac{1}{180H^2} (2\Psi_{k+3} - 27\Psi_{k+2} + 270\Psi_{k+1} - 490\Psi_k + 270\Psi_{k-1} - 27\Psi_{k-2} + 2\Psi_{k-3}), \\ \Psi'''(x_k) &\approx \frac{1}{8H^3} (-\Psi_{k+3} + 8\Psi_{k+2} - 13\Psi_{k+1} + 13\Psi_{k-1} - 8\Psi_{k-2} + \Psi_{k-3}), \\ \Psi^{(4)}(x_k) &\approx \frac{1}{6H^4} (-\Psi_{k+3} + 12\Psi_{k+2} - 39\Psi_{k+1} + 56\Psi_k - 39\Psi_{k-1} + 12\Psi_{k-2} - \Psi_{k-3}), \\ \Psi^{(5)}(x_k) &\approx \frac{1}{2H^5} (\Psi_{k+3} - 4\Psi_{k+2} + 5\Psi_{k+1} - 5\Psi_{k-1} + 4\Psi_{k-2} - \Psi_{k-3}), \\ \Psi^{(6)}(x_k) &\approx \frac{1}{H^6} (\Psi_{k+3} - 6\Psi_{k+2} + 15\Psi_{k+1} - 20\Psi_k + 15\Psi_{k-1} - 6\Psi_{k-2} + \Psi_{k-3}), \quad (2.71) \end{aligned}$$

$$\text{де } \Psi(x_k) = \Psi_k;$$

$$x_k = a + kH, \quad k = \overline{1, N-1};$$

$$H = L/N;$$

$$L = 1 - l_0/l_1.$$

З подання крайових умов (2.60) скінченними різницями другого порядку впливають співвідношення:

$$\Psi_0=0, \quad \Psi_{-2}=-\Psi_2, \quad \Psi_{-1}=-\Psi_1, \quad (2.72)$$

$$\Psi_N=0, \quad \Psi_{N+2}=-\Psi_{N-2}, \quad \Psi_{N+1}=-\Psi_{N-1}. \quad (2.73)$$

Після підстановки скінченних різниць (2.71) у рівняння (2.59) і виключення з них за допомогою (2.72), (2.73) змінних Ψ_s з номерами $s=-2, -1, 0, N, N+1, N+2$ виходить система однорідних алгебраїчних лінійних рівнянь $N-1$ порядку. Прирівнювання визначника цієї системи до нуля зумовлює характеристичне рівняння відносно параметрів хвилеутворення $\eta, \lambda, \chi, n, \gamma_1$.

Результатом застосування кожного з підходів є отримання характеристичного трансцендентного рівняння для відшукування критичної комбінації навантажень при мінімізації хвильового числа в окружному напрямі.

2.5.1.7 Порівняння результатів

Чисельному аналізу піддавалися оболонки з характеристиками $\alpha=60^\circ$, $E_1=E_2$, $\nu_1=\nu_2=\nu_3=\nu=0,3$, $l_1=182$ см, $l_0/l_1=0,65$, $h=0,5$ см, $h_1=h_2$ з урахуванням (2.45) і (2.47).

У таблиці 2.5 наведені значення безрозмірного критичного тиску χ , отримані трьома методами, за різних значень безрозмірних крутильного моменту λ і осевого зусилля η . Результати свідчать, що з двох аналітичних методів – ВКБ методу і ВКБ-Гальоркін методу, – найкраще узгодження з чисельним методом кінцевих різниць має ВКБ-Гальоркін метод.

Таблиця 2.5 – Порівняння результатів, отриманих різними методами

λ	η	n	χ		
			метод ВКБ-Гальоркін	метод скінченних різниць	ВКБ-метод
0,1	0	9	35,63027	35,61124	35,40195
	0,5	9	29,92725	29,92626	29,42852
1	0	9	35,32516	35,29868	34,90313
	0,5	9	29,5125	29,4943	29,17819
2	0	9	34,41797	34,37352	33,76563
	0,5	9	28,28516	28,2568	27,88398

2.5.1.8 Гранична поверхня, що відокремлює область стійкості від області нестійкості

Крайові умови для функції $\Psi(x)$ дозволяють отримати характеристичне трансцендентне рівняння

$$\Delta_1(\eta, \lambda, \chi, n, \gamma_1) = 0. \quad (2.74)$$

Для відшукування критичної комбінації навантажень, що задовольняють (2.74), необхідно знайти ті значення числа хвиль $n = n^*$ у окружному напрямі і тангенса кута нахилу $\gamma_1 = \gamma_1^*$ гребеня хвилі витріщеної хвилі до твірної, які відповідають найменшому значенню безрозмірного тиску $\chi = \chi_0$ при фіксованих значеннях безрозмірних крутильного моменту $\lambda = \lambda_0$ і стискаючого зусилля $\eta = \eta_0$. Множина точок, що задовольняють рівняння

$$\Delta_2(\eta_0, \lambda_0, \chi_0) = 0, \quad (2.75)$$

де $\Delta_2(\eta_0, \lambda_0, \chi_0) = \Delta_1(\eta_0, \lambda_0, \chi_0, n^*, \gamma_1^*)$, утворює граничну поверхню (поверхня стійкості). Ця поверхня відокремлює область стійкості від області нестійкості конічної тришарової оболонки.

На рис. 2.22 зображені гранична поверхня (2.75) оболонки (рис. 2.22, а), у якої $h_1 = h_2$, $\frac{h_3}{h} = 0,6$, $\frac{E_3}{E_1} = 0,01$, $\frac{l_0}{l_1} = 0,65$, її лінії рівня (рис. 2.22, б, в), а також погоджені з ними через рівняння (2.74) характеристики γ_1 нахилу гребеня витріщеної хвилі до твірної (рис. 2.22, в).

Аналізуючи результати, отримаємо висновок, що збільшення крутильного моменту дає зниження стійкості і збільшення кута нахилу витріщеної хвилі до твірної при заданих значеннях осьового зусилля.

Гібридним ВКБ-Гальоркін методом і за формулами, наведеними у монографії [317, с. 148], отримані залежності безрозмірного критичного тиску χ від h_3/h_1 . Оскільки в роботі [317] розглянута дія усебічного зовнішнього тиску без урахування впливу інших навантажень, то гібридний метод застосовувано за $\lambda = 0$, $\eta = 0$. Як впливає з порівняльного аналізу, результати, отримані в цій роботі, мають задовільне якісне і кількісне узгодження з результатами монографії [315], що свідчить про достовірність результатів цієї роботи.

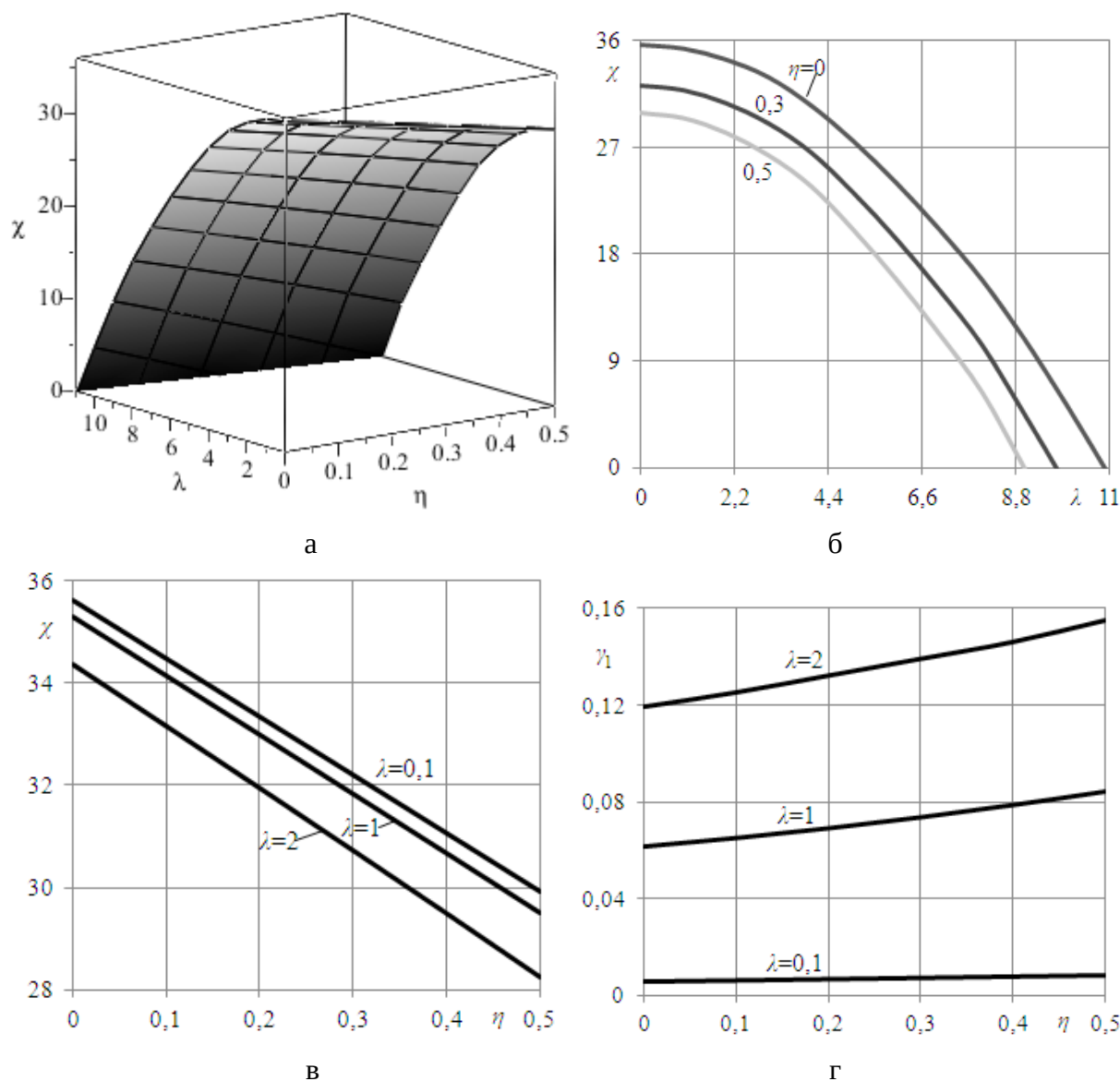


Рисунок 2.22 – Графіки граничної поверхні (а), її ліній рівня (б, в) і залежностей тангенса кута нахилу гребеня витріщеної хвилі до твірної оболонки y_1 від безрозмірної осьової сили

2.5.2 Вплив параметрів конічної тришарової оболонки на стійкість

Вплив параметрів конічної тришарової оболонки на стійкість досліджений при спільній дії на оболонку усебічного зовнішнього тиску, крутильного моменту і осьового стискання, залежно від h_3/h_1 (рис. 2.23–2.28). На кожному рис. 2.24, 2.26–2.28 безрозмірне значення крутильного моменту вибране рівним $\lambda=1$, при цьому, для ліній 1, 4 безрозмірне осьове зусилля рівне $\eta=0$, для ліній 2, 5 – $\eta=0,3$, ліній 3, 6 – $\eta=0,5$. Результати, наведені на рис. 2.24–2.26, відповідають оболонкам із зовнішніми шарами однакової товщини $h_1=h_2$, а лініям 4–6 на рис. 2.27, 2.28 – різною, де $h_1=2h_2$. На

рис. 2.24–2.27 значення відношення вибране $l_1/h=364$, а на рис. 2.28 для ліній 4–6 – $l_1/h=182$.

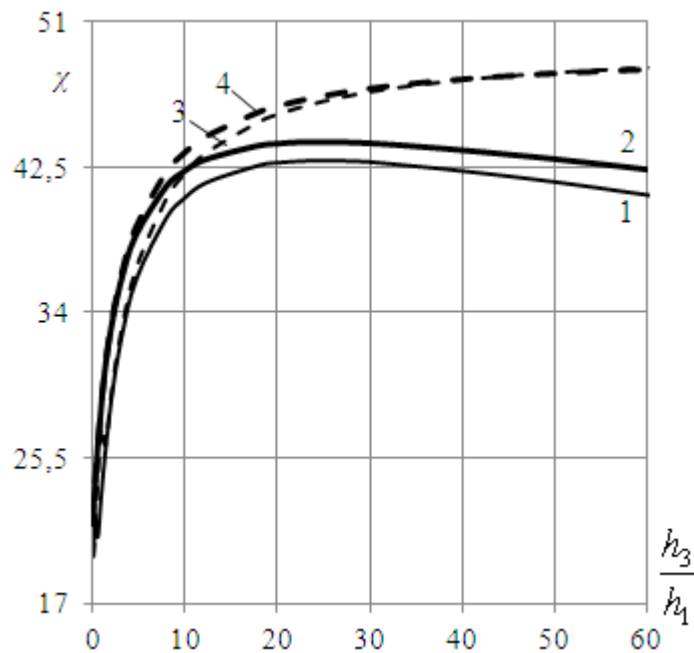


Рисунок 2.23 – Порівняльний аналіз результатів, отриманих гібридним ВКБ-Гальоркін

методом (лінії 2, 4) і за формулами з роботи [317] (лінії 1, 3), при $\frac{E_3}{E_1}=10^{-2}$ (лінії 1, 2) і

$\frac{E_3}{E_1}=10^{-4}$ (лінії 3, 4) для оболонки з параметром $\frac{l_0}{l_1}=0,65$

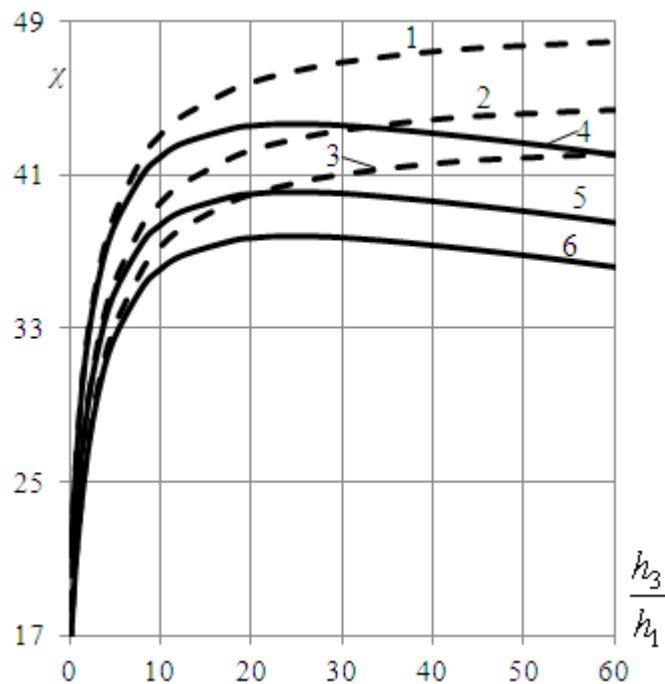


Рисунок 2.24 – Вплив відносної жорсткості ($E_3/E_1=10^{-4}$ (штрихові лінії 1–3); $E_3/E_1=10^{-2}$ (лінії 4–6)) і товщини внутрішнього шару h_3/h_1 на критичний тиск (при $l_0/l_1=0,65$)

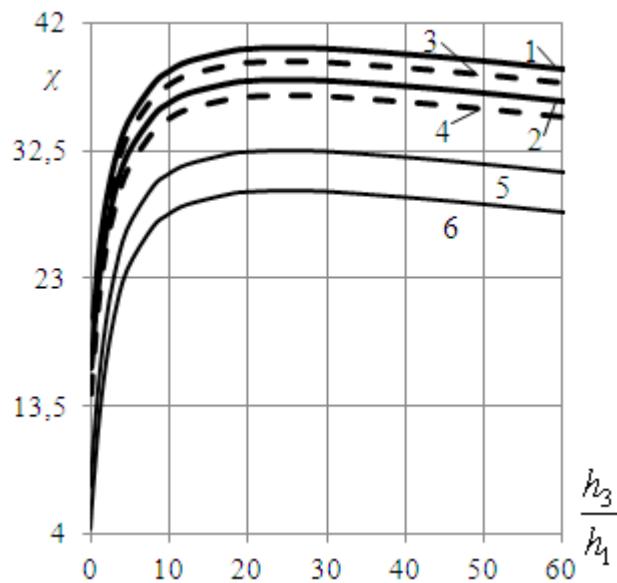


Рисунок 2.25 – Вплив безрозмірного крутильного моменту ($\lambda = 1$ (лінії 1, 2); $\lambda = 2$ (штрихові лінії 3, 4); $\lambda = 5$ (лінії 5, 6)) і осьового зусилля ($\eta = 0,3$ (лінії 1, 3, 5); $\eta = 0,5$ (лінії 2, 4, 6)) на критичний тиск оболонки з характеристиками $E_3/E_1 = 10^{-2}$, $l_0/l_1 = 0,65$

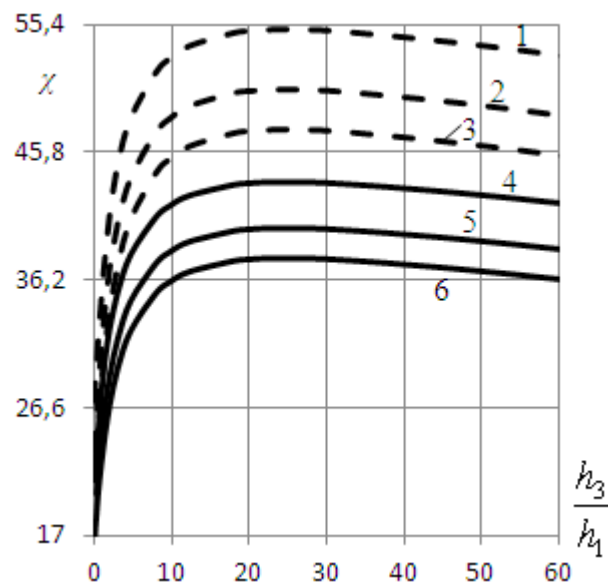


Рисунок 2.26 – Вплив відносної довжини твірної кінчної оболонки на критичний тиск ($L = 1 - l_0/l_1 = 0,25$ (штрихові лінії 1–3); $L = 0,35$ (суцільні лінії 4–6)) для оболонки з параметром $E_3/E_1 = 10^{-2}$

Кожна з побудованих залежностей (рис. 2.23–2.28) має точку локального максимуму, яка відповідає найбільш раціональній конструкції оболонки з точки зору стійкості. Цій точці, крім того, відповідає найменше значення тангенса кута нахилу гребеня хвилі витріщення до твірної. Це означає, що оболонка відповідної конструкції має більшу здатність чинити опір зсуву.

Аналіз отриманих результатів (рис. 2.24) доводить, що тришарова оболонка є стійкішою у разі меншого відносного модуля пружності E_2/E_1 . При

$E_3/E_1=10^{-2}$ найбільша стійкість відповідає оболонці з товщиною заповнювача, що становить близько 93% від загальної її товщини, при заданих значеннях λ , η , α , l_0/l_1 , l_1/h , h_2/h_1 (рис. 2.23–2.28); при $E_3/E_1=10^{-4}$ – близько 99% (рис. 2.24).

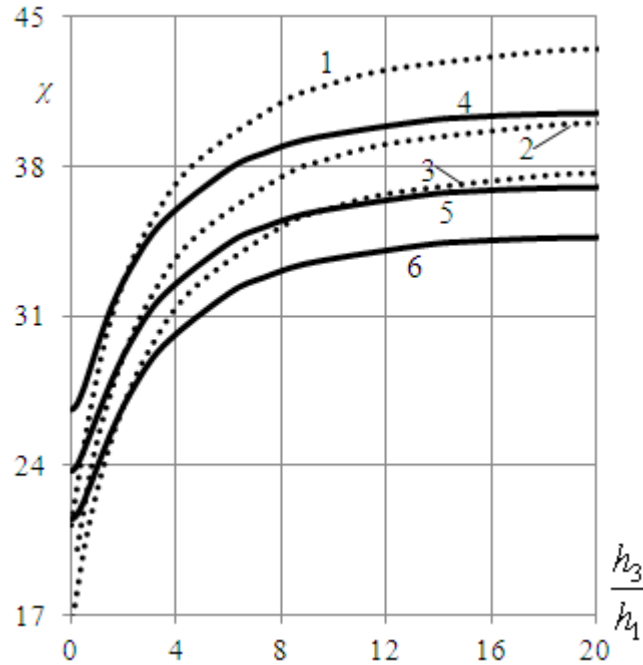


Рисунок 2.27 – Вплив товщини зовнішніх шарів тришарової оболонки на критичний тиск ($h_1=h_2$ (пунктирні лінії 1–3); $h_1=2h_2$ (лінії 4–6)) оболонки з характеристиками $E_3/E_1=10^{-2}$, $l_0/l_1=0,65$

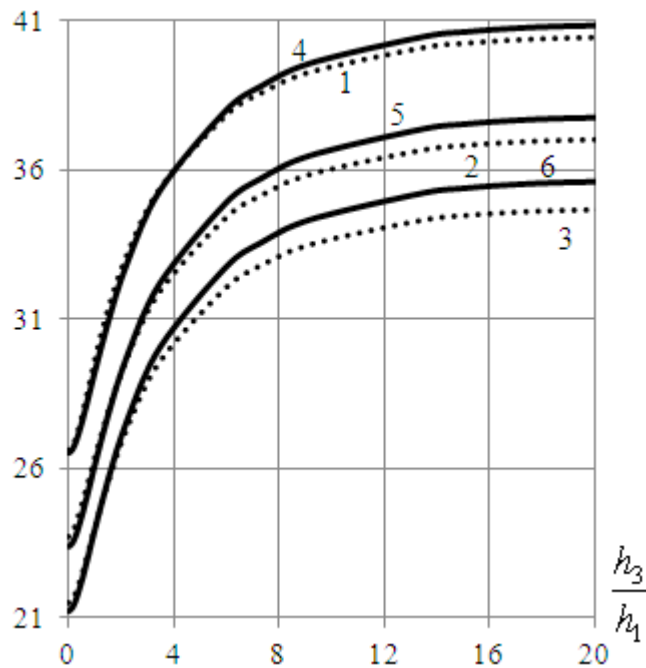


Рисунок 2.28 – Вплив товщини тришарової оболонки h відносно l_1 на критичний тиск ($l_1/h=364$ (пунктирні лінії 1–3); $l_1/h=182$ (лінії 4–6)) оболонки з параметрами $h_1=2h_2$, $E_3/E_1=10^{-2}$, $l_0/l_1=0,65$

2.5.3 Деформування циліндричних тришарових оболонок при дії зовнішнього тиску

Розглянемо задачу визначення напружено-деформованого стану циліндричної тришарової оболонки при дії рівномірного зовнішнього тиску (рис. 2.29).

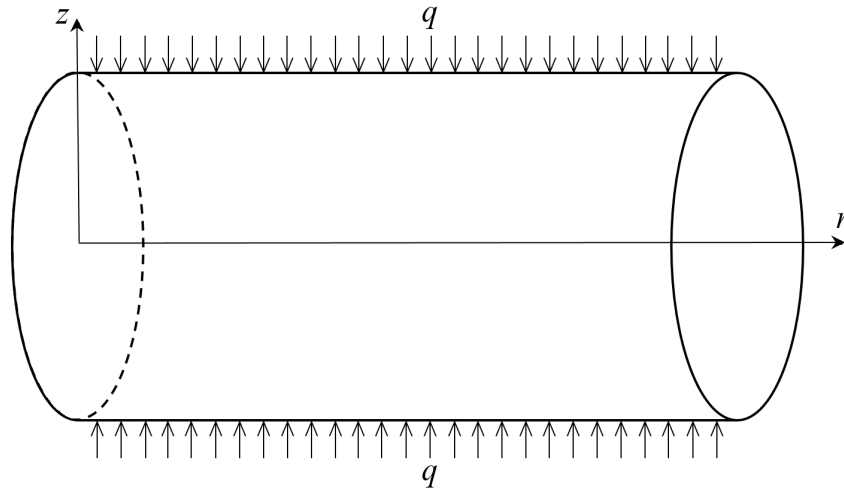


Рисунок 2.29 – Циліндрична оболонка під дією рівномірного зовнішнього тиску

Параметри Ламе для циліндричної оболонки приймуть значення $A_1=1$ та $A_2=R$, а головні кривини – $k_1=0$, $k_2=\frac{1}{R}$. Вважаємо далі, що $u_1(r)$ – переміщення серединної поверхні заповнювача, $u_3(r)$ – переміщення вздовж поперечної координати, $\psi_1(r)$ – кут повороту нормалі до серединної поверхні заповнювача у площині rz .

Функціонал повної потенціальної енергії тришарової циліндричної оболонки при дії рівномірного тиску має вигляд:

$$\begin{aligned}
 E = & \iint \left[\frac{1}{2} \int_0^{h^{(1)}} (\sigma_r^{(1)} \varepsilon_r^{(1)} + \sigma_\varphi^{(1)} \varepsilon_\varphi^{(1)}) \delta_1^{(1)} \delta_2^{(1)} dz + \right. \\
 & + \frac{1}{2} \int_{h^{(1)}+h^{(3)}}^{h^{(1)}+h^{(2)}+h^{(3)}} (\sigma_r^{(2)} \varepsilon_r^{(2)} + \sigma_\varphi^{(2)} \varepsilon_\varphi^{(2)}) \delta_1^{(2)} \delta_2^{(2)} dz + \\
 & \left. + \frac{1}{2} \int_{h^{(1)}}^{h^{(1)}+h^{(3)}} (\sigma_r^{(3)} \varepsilon_r^{(3)} + \sigma_\varphi^{(3)} \varepsilon_\varphi^{(3)} + \tau_{rz}^{(3)} \varepsilon_{rz}^{(3)}) \delta_1^{(2)} \delta_3^{(3)} dz \right] R dr d\varphi - A = 0.
 \end{aligned} \tag{2.76}$$

Робота зовнішніх сил у (2.71) для задачі деформування запишеться у такому вигляді [330]:

$$A = \iint q u_3(r, \varphi) \left(\frac{h^{(1)} + h^{(3)}}{R} + 1 \right) R dr d\varphi.$$

Граничні умови для випадку защемлення кругових контурів оболонки можна виразити через переміщення $u_1(r)$, $u_3(r)$, $\psi_1(r)$ [331]:

$$u_3(r) = u_1(r) = \psi_1(r) = 0, \quad r = 0, L. \quad (2.77)$$

Після підстановки виразів компонентів деформації $\varepsilon_r^{(i)}$, $\varepsilon_\varphi^{(i)}$, $\varepsilon_{rz}^{(3)}$ та напружень $\sigma_r^{(i)}$, $\sigma_\varphi^{(i)}$, $\tau_{rz}^{(3)}$, $i = \overline{1, 3}$, які наведені вище, у рівнянні (2.76) отримаємо вираз повної потенційної енергії через шукані переміщення $u_1(r)$, $u_3(r)$, $\psi_1(r)$.

Таким чином, вираз (2.76) та граничні умови (2.77) замикають задачу визначення напружено-деформованого стану циліндричної оболонки при дії стискаючого навантаження. Для розв'язання даної задачі використаємо метод Рітца. Координатні функції, що відповідають граничним умовам (2.77), матимуть вигляд:

$$u_1(r) = \sum_{i=1}^n A_i \sin\left(\frac{i\pi r}{L}\right), \quad u_3(r) = \sum_{i=1}^n B_i \sin\left(\frac{i\pi r}{L}\right), \quad \psi_1(r) = \sum_{i=1}^n C_i \sin\left(\frac{i\pi r}{L}\right). \quad (2.78)$$

Підставляючи (2.78) у функціонал енергії (2.76) та обчислюючи похідні

$$\frac{\partial E}{\partial A_i}, \quad \frac{\partial E}{\partial B_i}, \quad \frac{\partial E}{\partial C_i}, \quad i = \overline{1, n},$$

отримаємо системи лінійних або нелінійних алгебраїчних рівнянь, які відповідають лінійно-пружній (за Г. Каудерером [332]) та нелінійно-пружній постановці задачі.

Як частинний випадок розглянемо вісесиметричну деформацію одношарової циліндричної оболонки під дією зовнішнього тиску та порівняємо отримані значення переміщень та напружень зі значеннями, отриманими за скінченно-елементною моделлю. Параметри оболонки: довжина $L = 4$ м, радіус $R = 2$ м, товщина $h = 0,037$ м, матеріал – сталь.

На наведених нижче рисунках суцільною лінію позначено розрахунки за аналітичною моделлю, а пунктиром – за скінченно-елементною моделлю. Так, на рис. 2.30 зображено переміщення $u_3(r)$ при дії навантаження $q = 0,05$ МПа та інтенсивність напружень в одношаровій циліндричній оболонці для даної постановки задачі.

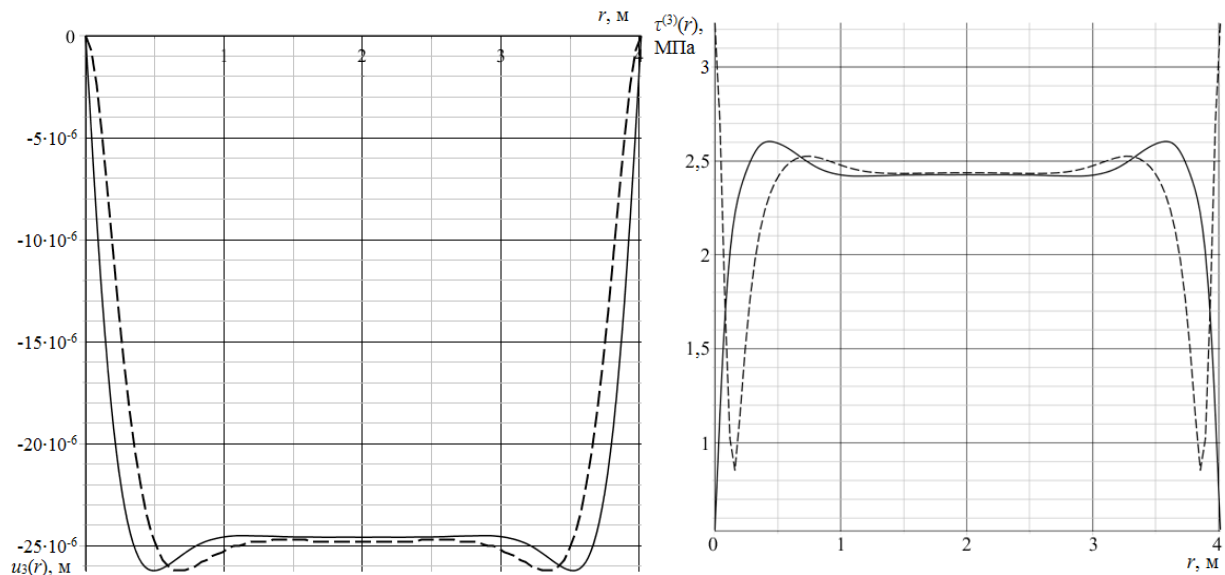


Рисунок 2.30 – Переміщення $u_3(r)$ одношарової циліндричної оболонки під дією рівномірного зовнішнього тиску та інтенсивність напружень одношарової циліндричної оболонки під дією рівномірного зовнішнього тиску $\tau^3(r)$

Максимальні за модулем значення прогину, згідно розробленої аналітичної моделі, досягаються при значенні $r=0,5$ м, $r=3,5$ м та дорівнюють $u_3 = -26,5 \cdot 10^{-6}$ м.

Відхилення наведених значень переміщень за розробленою моделлю від скінченно-елементної моделі пояснюються тим, що при побудові цих моделей використовувались відмінні спрощуючі гіпотези. Так, при побудові аналітичної моделі одношарової циліндричної оболонки вважалась справедливою гіпотеза Кірхгофа-Лява. На відміну від цього, при побудові скінченно-елементної моделі використовувалися співвідношення теорії Міндліна для оболонок. Цим же пояснюється якісна відмінність функції інтенсивності напружень у випадку аналітичної моделі та скінченно-елементної моделі (рис. 2.30).

Розглянемо далі аналогічну задачу для тришарової циліндричної оболонки з наступними параметрами: товщина обшивок $h^{(1)}=h^{(2)}=h=0,001$ м, матеріал обшивок – сталь; товщина заповнювача $h^{(3)}=0,035$ м, матеріал заповнювача – Д16Т, довжина та радіус оболонки $L=4$ м, $R=2$ м відповідно.

Розглянемо вплив нелінійної пружності матеріалу заповнювача на величини переміщень та інтенсивності напружень. На рис. 2.31 зображені переміщення $u_3(r)$ та інтенсивність напружень $\tau^{(3)}$ тришарових циліндричних оболонок відповідно.

На рис. 2.32 наведено значення переміщення $u_3(r)$ та інтенсивності напружень $\tau^{(3)}$ при врахуванні нелінійної пружності матеріалу заповнювача тришарової циліндричної оболонки (розв'язок для нелінійно-пружного випадку позначено пунктиром).

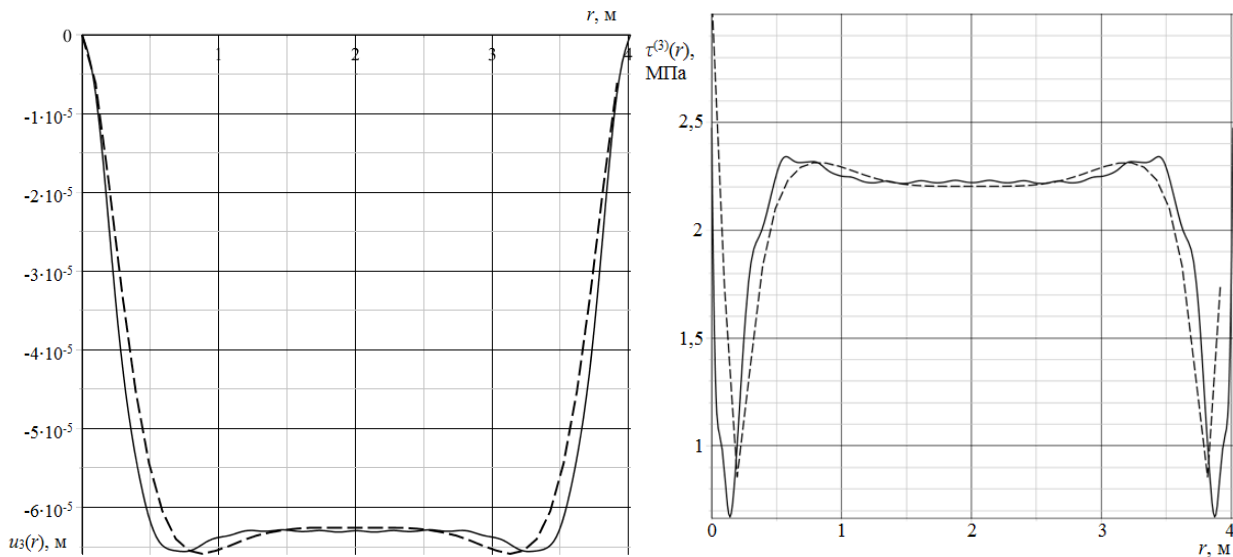


Рисунок 2.31 – Прогин тришарової циліндричної оболонки під дією рівномірного зовнішнього тиску та інтенсивність напружень тришарової циліндричної оболонки під дією рівномірного зовнішнього тиску

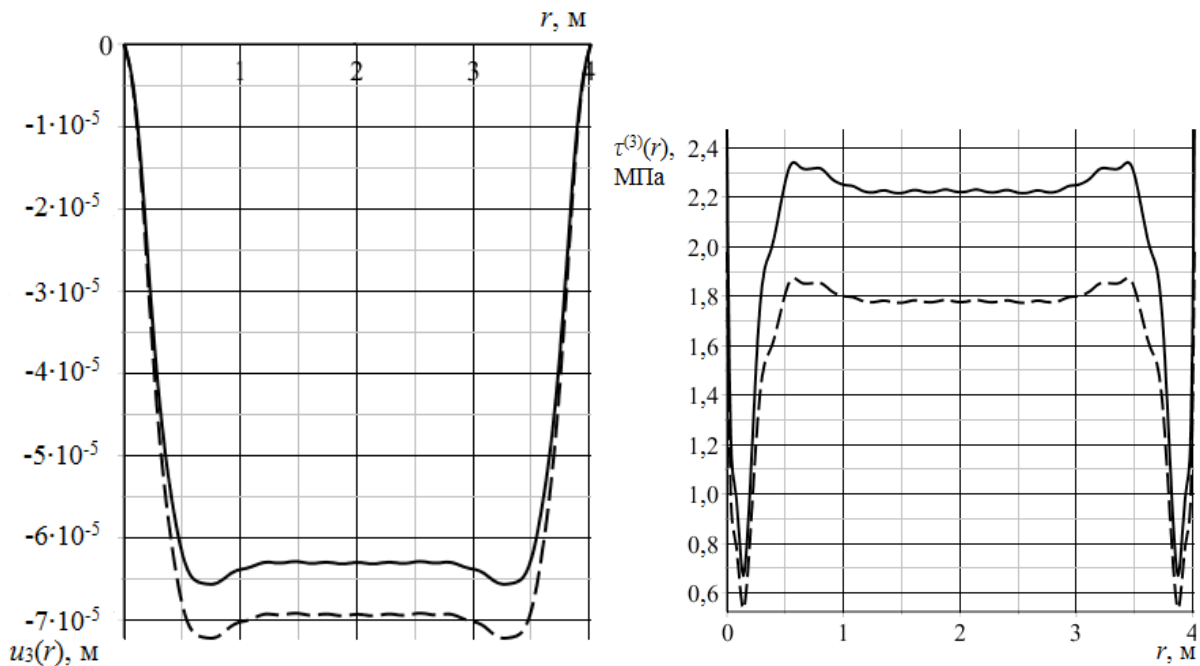


Рисунок 2.32 – Переміщення тришарової циліндричної оболонки з нелінійно-пружним заповнювачем під дією рівномірного зовнішнього тиску та інтенсивність напружень циліндричної оболонки з нелінійно-пружним заповнювачем під дією рівномірного зовнішнього тиску

З порівняння рис. 2.32 можна зробити висновок, що переміщення $u_3(r)$, збільшуються на 10%–20% при врахуванні нелінійної пружності заповнювача. Натомість інтенсивність напружень $\tau^{(3)}$ зменшується в середньому на 10%. Такі результати в цілому якісно відповідають результатам які були отримані при аналогічному порівнянні для круглої пластини.

РОЗДІЛ 3. ПОСТПРОЦЕСОР

3.1. Метод генерації адаптивних дискретних моделей форм об'єктів, заданих функціонально

Загальна ідея генерації адаптивних дискретних моделей форм об'єктів полягає розбитті певних елементів шляхом додавання нових вузлів за результатами аналізу деякої величини, яка визначена у кожному вузлі моделі.

Нехай для форми об'єкта Ω , що задана функцією F , побудована дискретна модель M і множина кодів елементів V , які необхідно розбити. Техніку додавання нових вузлів у центри описаних кіл (або сфер у тривимірному випадку) можна використовувати для згущення дискретних моделей форм об'єктів на базі симплексів. Результатом того, що нові вузли додаються у центри описаних кіл, є виконання критерію Делоне на кожному кроці адаптації. Проте, ця техніка практично не застосовна для чотирикутників і шестигранників.

З точки зору розробки узагальнених методів генерації дискретних моделей форм об'єктів перспективною є розробка шаблонів і правил розбиття кожного елементу моделі та відновлення коректності топології дискретної моделі шляхом розбиття сусідніх елементів.

Наприклад, на першому етапі кожен трикутник можна розбити на три нових додаванням вузлів у середини його сторін (рис. 3.1). Далі кожен трикутник, у якого є спільне ребро з одним із розбитих на першому етапі, розбивається на дві частини шляхом додавання нового вузла у середину цього ребра. Як результат отримано два можливих шаблони розбиття. Ця схема відома під назвою «зелені трикутники» [333]. Така схема перетворення трикутників дозволяє рівномірно генерувати дискретні моделі форм об'єктів на базі трикутних елементів.

Схему «зелені трикутники» можна узагальнити на випадок тетраедрів. На першому етапі кожний обраний тетраедр можна розбити на вісім елементів додаванням нових вузлів у середини його сторін (рис. 3.2–3.3). Далі, якщо у сусіднього тетраедра є спільне ребро з тетраедром, розбитим на першому етапі, то він розбивається на два елементи шляхом додавання нового вузла у середину цього ребра (рис. 3.3, а).

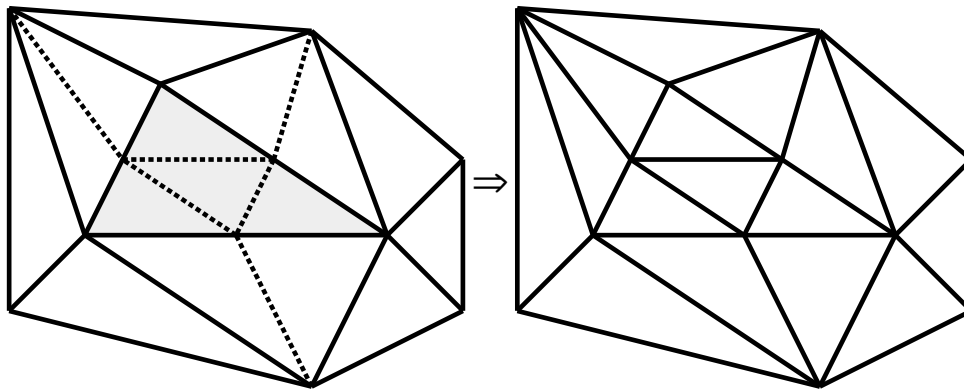


Рисунок 3.1 – Схема розбиття трикутних елементів

Якщо у тетраедра, сусіднього з тетраедром, розбитим на першому етапі, є спільна грань, то він розбивається на чотири частини шляхом додавання нових вузлів у середини ребер, які належать спільній грані (рис. 3.3, б). Отже, для тетраедрів отримаємо три шаблони розбиття.

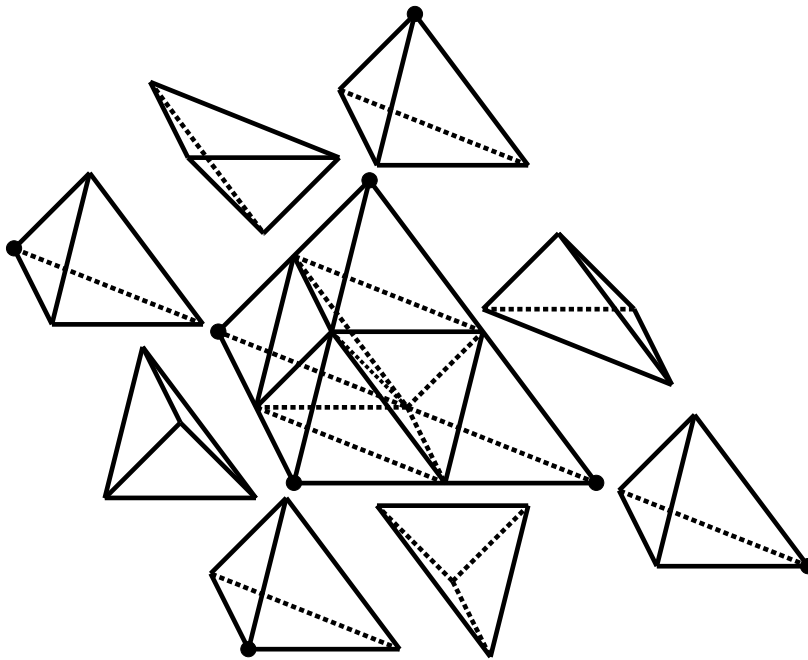


Рисунок 3.2 – Схема розбиття тетраедричних елементів

Для дискретних моделей на базі чотирикутних елементів на першому етапі можна використовувати схему розбиття елемента на дев'ять частин (рис. 3.4) [214]. Водночас для сусідніх елементів на другому етапі буде достатньо двох додаткових шаблонів. Якщо у сусіднього елемента є ребро, спільне з розбитим на першому етапі, то він розбивається на чотири частини. Якщо у чотирикутника є два суміжних ребра, які спільні з розбитим на першому етапі, то він розбивається на п'ять частин.

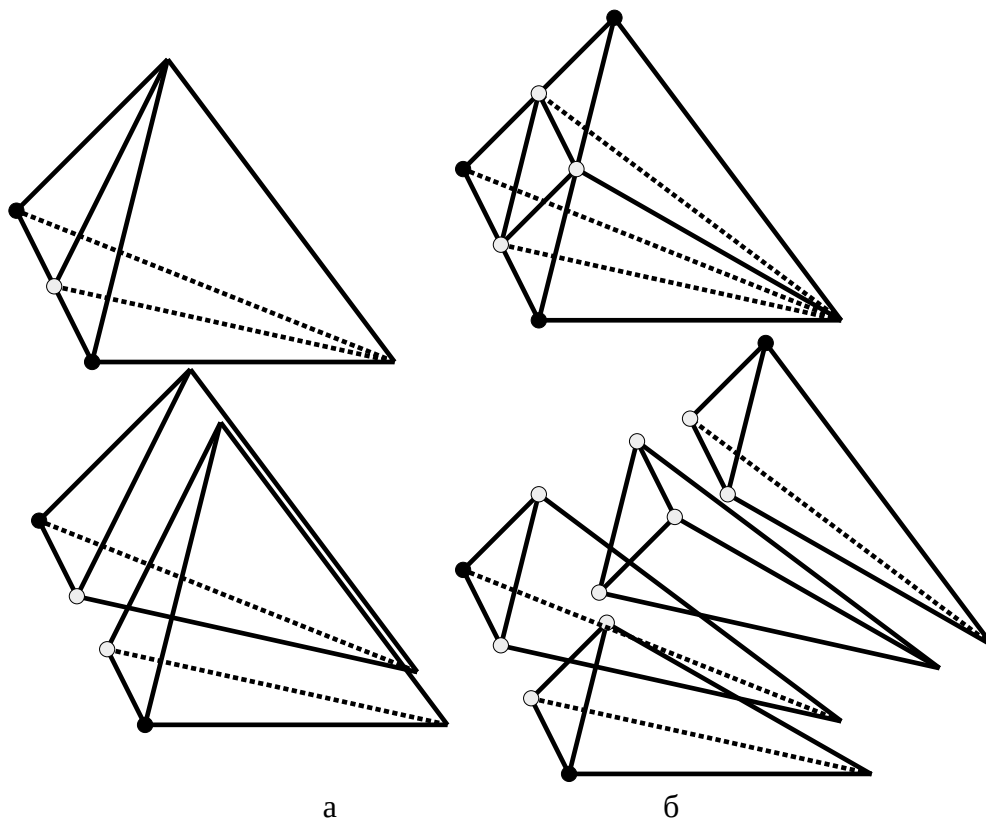


Рисунок 3.3 – Додаткові шаблони розбиття тетраедричних елементів:
а – спільне ребро; б – спільна грань

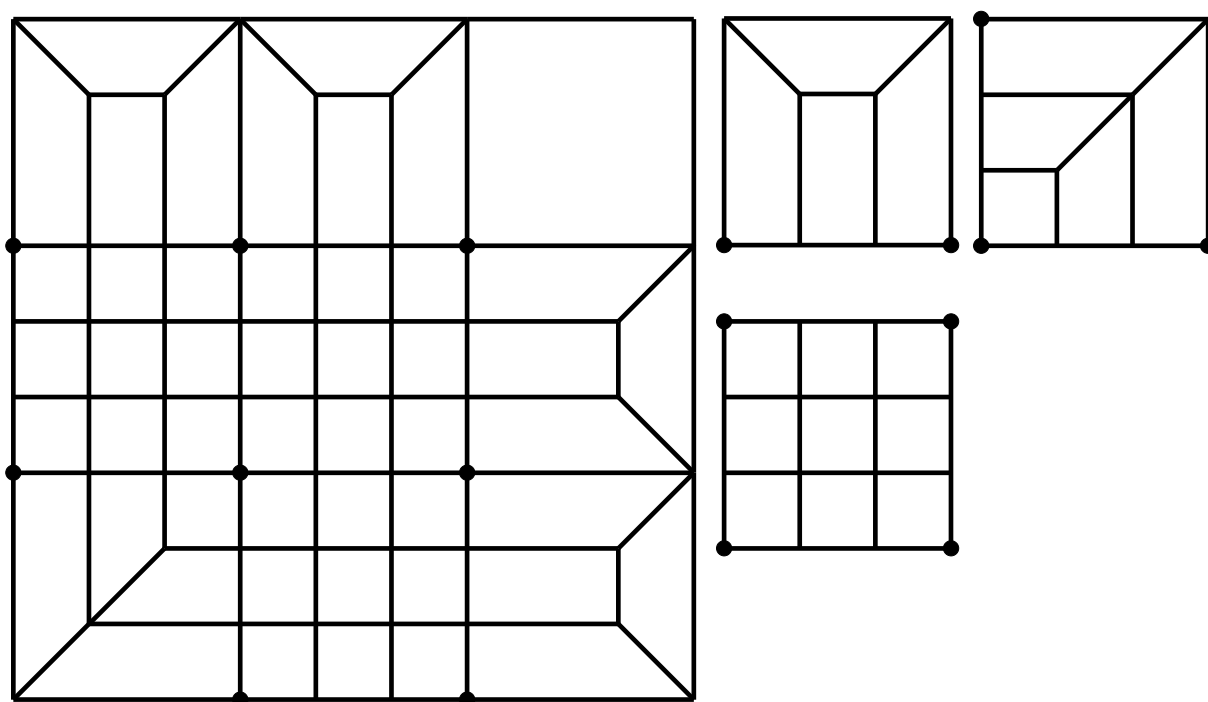


Рисунок 3.4 – Схема розбиття чотирикутних елементів

У випадку дискретних моделей на базі шестигранних елементів на першому етапі кожен шестигранник розбиваємо на 27 частин (рис. 3.5) [214]. На наступному етапі, якщо у деякого елемента ребро є учасником розбиття, то він розбивається на п'ять частин. Якщо у шестигранника одна грань є учасником

розбиття, то його розбиваємо на тринадцять частин. Якщо у розбитті беруть участь три вершини, які належать до однієї грані, то такий шестигранник розбиваємо на шість частин. Якщо у розбитті беруть участь шість вершин, які належать двом суміжним граням, то такий елемент розбиваємо на п'ять частин. У двох останніх випадках для нових елементів необхідно додатково застосувати процедуру перевірки необхідності додаткового розбиття.

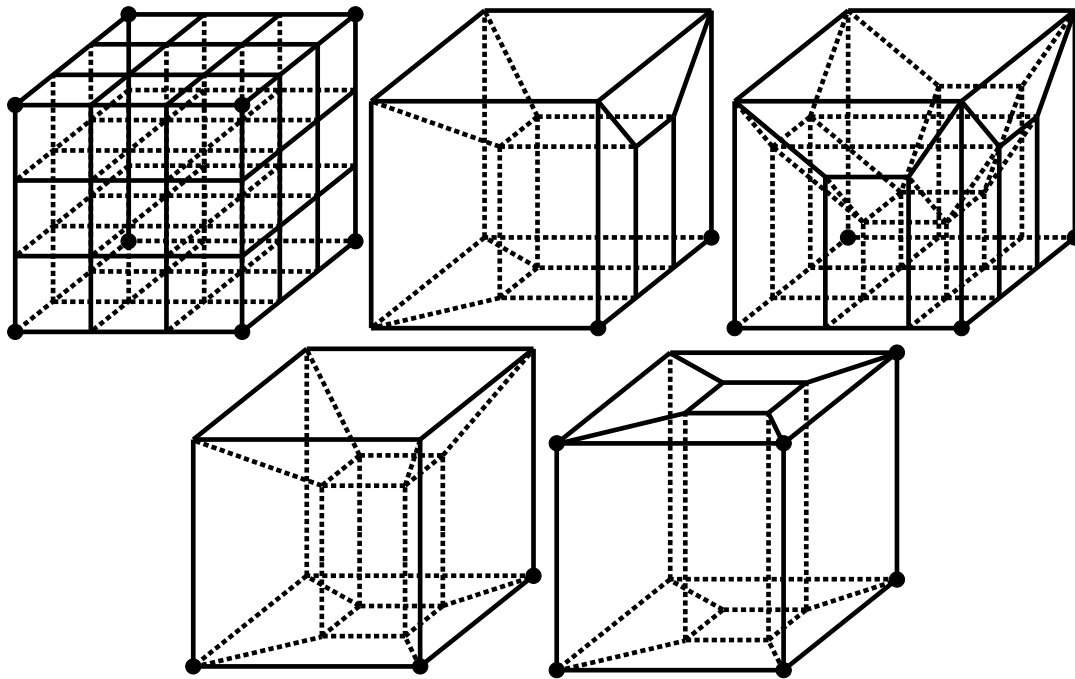


Рисунок 3.5 – Схема розбиття шестигранних елементів

Отже, для кожної найбільш поширеної форми елемента визначені основні схеми розбиття. Всього їх з урахуванням обертання для трикутників – 8, тетраедрів і чотирикутників – 16, шестигранників – 256. Якщо кожному кожному вузлу, що бере участь у розбитті, буде відповідати код 1, а всім іншим – 0, то кожному шаблону розбиття буде відповідати двійковий код. У результаті можна використовувати такий алгоритм побудови розбиття:

algorithm subdivide

input:

F – функція, що подає форму об'єкта Ω

M – дискретна модель

V – множина елементів для розбиття

ε – чисельний нуль

begin

$E \leftarrow \emptyset$

$N \leftarrow \emptyset$

for each $k_e \in V$ **do**

$e \leftarrow M(k_e)$

```

for each  $k_n \in e$  do
     $N \leftarrow N \cup k_n$ 
     $E \leftarrow E \cup M.\text{incident}(k_n)$ 
end for
end for
for each  $k_e \in E$  do
     $e \leftarrow M(k_e)$ 
     $code \leftarrow 0$ 
    for  $i \leftarrow 1$  to  $|e|$  do
        if  $e[i] \in N$  then
             $code \leftarrow code \mid 2^{i-1}$ 
        end if
    end for
     $S \leftarrow \text{template}(M.V(e), code)$ 
    for each  $s \in S$  do
         $\text{insert\_element}(M, s, \varepsilon)$ 
    end for
     $\text{remove\_element}(M, k_e)$ 
end for
for each  $n \in \delta(M.N)$  do
    if  $F(n.v) \geq \varepsilon$  then
         $h \leftarrow \min \|n.v - \text{neighbors}(M, M.\text{code}(n))\|$ 
         $n.v \leftarrow \text{find\_border}(n.v, F, h, \varepsilon)$ 
    end if
end for
end

```

Отже, на першому кроці алгоритму формуємо дві множини: E і N – відповідно, множини елементів і вузлів, які беруть участь у розбитті. Множина N складається з кодів всіх вузлів, які належать елементам з вихідної множини V , а множина E – з елементів, інцидентним у вузлах N .

На другому кроці алгоритму для кожного елемента з E визначаємо його двійковий код: якщо вузол у вершині бере участь у розбитті, то у позиції, визначеній порядком обходу вершин елемента, ставимо 1, інакше – 0. Далі за допомогою алгоритму `template` формуємо нові елементи.

На третьому етапі поновлюємо координати межових вузлів. Для цього обчислюємо проекції на межу Ω вузлів, для яких функція F не рівна 0.

Вигляд алгоритму для формування нових елементів залежить від форми елемента. Наприклад, для трикутного елемента (порядок обходу основних і додаткових вузлів наведено на рис. 3.6, а), він матиме такий вигляд:

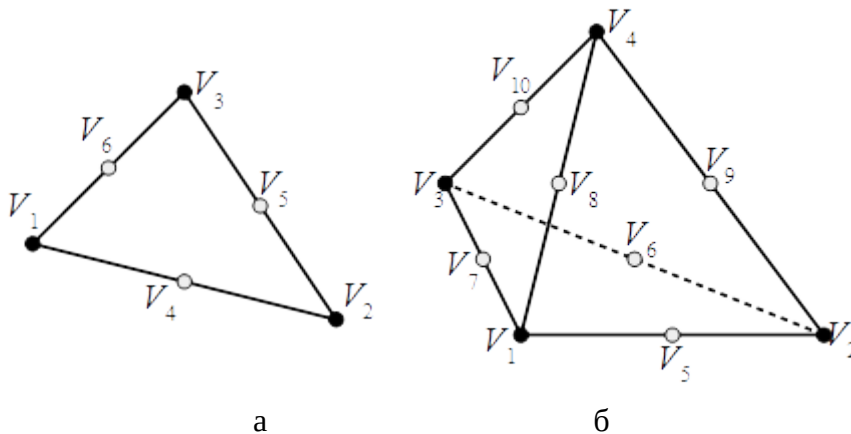
algorithm template**input:** V – список з трьох вершин трикутника $code$ – код шаблона**output:** E_{code} – множина елементів, які відповідають коду шаблона**begin** $V_4 \leftarrow 0,5 (V_1 + V_2)$ $V_5 \leftarrow 0,5 (V_2 + V_3)$ $V_6 \leftarrow 0,5 (V_3 + V_1)$ $E_0 \leftarrow \{(V_1, V_2, V_3)\}$ $E_1 \leftarrow \{(V_1, V_2, V_3)\}$ $E_2 \leftarrow \{(V_1, V_2, V_3)\}$ $E_3 \leftarrow \{(V_1, V_4, V_3), (V_2, V_3, V_4)\}$ $E_4 \leftarrow \{(V_1, V_2, V_3)\}$ $E_5 \leftarrow \{(V_1, V_2, V_6), (V_3, V_6, V_2)\}$ $E_6 \leftarrow \{(V_2, V_5, V_1), (V_3, V_1, V_5)\}$ $E_7 \leftarrow \{(V_1, V_4, V_6), (V_2, V_5, V_4), (V_3, V_6, V_5), (V_4, V_5, V_6)\}$ **return** E_{code} 

Рисунок 3.6 – Нумерація основних і вузлів у симплексах: а – трикутник; б – тетраедр

Для тетраедрів порядок обходу основних і додаткових вузлів зображено на рис. 3.6, б. У результаті алгоритму формування нових елементів отримає такий вигляд:

algorithm template**input:** V – список з чотирьох вершин тетраедра $code$ – код шаблона**output:** E_{code} – множина елементів, які відповідають коду шаблона**begin**


```

 $V_5 \leftarrow 0,5 (V_1 + V_2)$ 
 $V_6 \leftarrow 0,5 (V_2 + V_3)$ 
 $V_7 \leftarrow 0,5 (V_3 + V_1)$ 
 $V_8 \leftarrow 0,5 (V_1 + V_4)$ 
 $V_9 \leftarrow 0,5 (V_2 + V_4)$ 
 $V_{10} \leftarrow 0,5 (V_3 + V_4)$ 
 $E_0 \leftarrow \{(V_1, V_2, V_3, V_4)\}$ 
 $E_1 \leftarrow \{(V_1, V_2, V_3, V_4)\}$ 
 $E_2 \leftarrow \{(V_1, V_2, V_3, V_4)\}$ 
 $E_3 \leftarrow \{(V_1, V_5, V_3, V_4), (V_2, V_3, V_5, V_4)\}$ 
 $E_4 \leftarrow \{(V_1, V_2, V_3, V_4)\}$ 
 $E_5 \leftarrow \{(V_1, V_2, V_7, V_4), (V_3, V_7, V_2, V_4)\}$ 
 $E_6 \leftarrow \{(V_2, V_6, V_1, V_4), (V_3, V_1, V_6, V_4)\}$ 
 $E_7 \leftarrow \{(V_1, V_5, V_7, V_4), (V_2, V_6, V_5, V_4), (V_3, V_7, V_6, V_4), (V_5, V_6, V_7, V_4)\}$ 
 $E_8 \leftarrow \{(V_1, V_2, V_3, V_4)\}$ 
 $E_9 \leftarrow \{(V_1, V_2, V_3, V_8), (V_8, V_2, V_3, V_4)\}$ 
 $E_{10} \leftarrow \{(V_1, V_2, V_3, V_9), (V_1, V_9, V_3, V_4)\}$ 
 $E_{11} \leftarrow \{(V_1, V_8, V_5, V_3), (V_2, V_5, V_9, V_3), (V_4, V_9, V_8, V_3), (V_5, V_8, V_9, V_3)\}$ 
 $E_{12} \leftarrow \{(V_1, V_2, V_3, V_{10}), (V_1, V_2, V_{10}, V_4)\}$ 
 $E_{13} \leftarrow \{(V_1, V_7, V_8, V_2), (V_3, V_{10}, V_7, V_2), (V_4, V_8, V_{10}, V_2), (V_7, V_{10}, V_8, V_2)\}$ 
 $E_{14} \leftarrow \{(V_2, V_9, V_6, V_1), (V_3, V_6, V_{10}, V_1), (V_4, V_{10}, V_9, V_1), (V_6, V_9, V_{10}, V_1)\}$ 
 $E_{15} \leftarrow \{(V_1, V_5, V_7, V_8), (V_2, V_6, V_5, V_9), (V_3, V_7, V_6, V_{10}), (V_8, V_9, V_{10}, V_4),$ 
 $(V_5, V_9, V_7, V_8), (V_7, V_9, V_{10}, V_8), (V_5, V_7, V_9, V_6), (V_7, V_{10}, V_9, V_6)\}$ 
return  $E_{code}$ 

```

Для чотирикутних елементів основні та додаткові вузли можливо перенумерувати з використанням номерів рядків і стовпців, на яких вони розташовані (рис. 3.7). У результаті алгоритм формування нових елементів отримає такий вигляд:

algorithm template

input:

V – список з чотирьох вершин чотирикутника

$code$ – код шаблона

output:

E_{code} – множина елементів, які відповідають коду шаблона

begin

$$\xi_j \leftarrow -1 + \frac{2}{3}j, \quad j = \overline{0, 3}$$

$$\eta_i \leftarrow -1 + \frac{2}{3}i, \quad i = \overline{0, 3}$$

$$v_{ij} \leftarrow \frac{1}{4} [V_1(1 - \xi_j)(1 - \eta_i) + V_2(1 + \xi_j)(1 - \eta_i) + V_3(1 + \xi_j)(1 + \eta_i) + V_4(1 - \xi_j)(1 + \eta_i)]$$

$$E_0 \leftarrow \{(v_{00}, v_{03}, v_{33}, v_{30})\}$$

$$\begin{aligned}
\mathbf{E}_1 &\leftarrow \{(V_{00}, V_{03}, V_{33}, V_{30})\} \\
\mathbf{E}_2 &\leftarrow \{(V_{00}, V_{03}, V_{33}, V_{30})\} \\
\mathbf{E}_3 &\leftarrow \{(V_{00}, V_{03}, V_{12}, V_{10}), (V_{10}, V_{12}, V_{22}, V_{20}), (V_{20}, V_{22}, V_{33}, V_{30}), (V_{12}, V_{03}, V_{33}, V_{22})\} \\
\mathbf{E}_4 &\leftarrow \{(V_{00}, V_{03}, V_{33}, V_{30})\} \\
\mathbf{E}_5 &\leftarrow \{(V_{00}, V_{03}, V_{33}, V_{30})\} \\
\mathbf{E}_6 &\leftarrow \{(V_{00}, V_{03}, V_{12}, V_{11}), (V_{00}, V_{11}, V_{31}, V_{30}), (V_{31}, V_{11}, V_{12}, V_{32}), (V_{32}, V_{12}, V_{03}, V_{33})\} \\
\mathbf{E}_7 &\leftarrow \{(V_{00}, V_{03}, V_{12}, V_{10}), (V_{10}, V_{12}, V_{21}, V_{20}), (V_{20}, V_{21}, V_{31}, V_{30}), (V_{21}, V_{12}, V_{32}, V_{31}), \\
&\quad (V_{12}, V_{03}, V_{33}, V_{32})\} \\
\mathbf{E}_8 &\leftarrow \{(V_{00}, V_{03}, V_{33}, V_{30})\} \\
\mathbf{E}_9 &\leftarrow \{(V_{00}, V_{01}, V_{21}, V_{30}), (V_{01}, V_{02}, V_{22}, V_{21}), (V_{02}, V_{03}, V_{33}, V_{22}), (V_{30}, V_{21}, V_{22}, V_{33})\} \\
\mathbf{E}_{10} &\leftarrow \{(V_{00}, V_{03}, V_{33}, V_{30})\} \\
\mathbf{E}_{11} &\leftarrow \{(V_{00}, V_{01}, V_{11}, V_{10}), (V_{01}, V_{02}, V_{22}, V_{11}), (V_{02}, V_{03}, V_{33}, V_{22}), (V_{10}, V_{11}, V_{22}, V_{20}), \\
&\quad (V_{20}, V_{22}, V_{33}, V_{30})\} \\
\mathbf{E}_{12} &\leftarrow \{(V_{00}, V_{03}, V_{13}, V_{11}), (V_{11}, V_{13}, V_{23}, V_{21}), (V_{21}, V_{23}, V_{33}, V_{30}), (V_{00}, V_{11}, V_{21}, V_{30})\} \\
\mathbf{E}_{13} &\leftarrow \{(V_{00}, V_{01}, V_{21}, V_{30}), (V_{01}, V_{02}, V_{12}, V_{21}), (V_{02}, V_{03}, V_{13}, V_{12}), (V_{21}, V_{12}, V_{13}, V_{23}), \\
&\quad (V_{30}, V_{21}, V_{23}, V_{33})\} \\
\mathbf{E}_{14} &\leftarrow \{(V_{00}, V_{03}, V_{13}, V_{11}), (V_{11}, V_{13}, V_{23}, V_{22}), (V_{22}, V_{23}, V_{33}, V_{32}), (V_{32}, V_{31}, V_{11}, V_{22}), \\
&\quad (V_{00}, V_{11}, V_{31}, V_{30})\} \\
\mathbf{E}_0 &\leftarrow \{(V_{00}, V_{01}, V_{11}, V_{10}), (V_{10}, V_{11}, V_{21}, V_{20}), (V_{20}, V_{21}, V_{31}, V_{30}), (V_{01}, V_{02}, V_{12}, V_{11}), \\
&\quad (V_{11}, V_{12}, V_{22}, V_{21}), (V_{21}, V_{22}, V_{32}, V_{31}), (V_{02}, V_{03}, V_{13}, V_{12}), (V_{12}, V_{13}, V_{23}, V_{22}), \\
&\quad (V_{22}, V_{23}, V_{33}, V_{32})\}
\end{aligned}$$

return $\mathbf{E}_{code}$

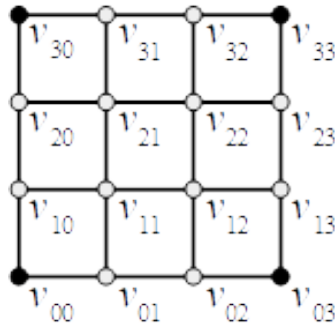


Рисунок 3.7 – Нумерація основних і додаткових вузлів у чотирикутниках

Для шестигранних елементів кількість можливих шаблонів дорівнює 256. Через велику довжину списку елементів алгоритм формування нових шестигранних елементів не приведено. Програмний код, який реалізує шаблони для шестигранних елементів, можна знайти у репозиторії за таким посиланням: <https://github.com/qzcad/qzcad-tree/blob/master/mesh/hexahedralmesh3d.cpp>

3.2. Метод формування множини елементів для розбиття

Як зазначено у підрозділі 2.1, для знаходження значень невідомої неперервної величини у вузлах дискретної моделі форми об'єкта будують систему лінійних алгебраїчних рівнянь, матрицю якої називають глобальною матрицею. Процес побудови цієї матриці називають ансамблюванням. Ансамблювання є спеціальною сумою локальних матриць, кожна з яких є результатом інтегрування вихідного диференціального рівняння на скінченному елементі. Розмір глобальної матриці дорівнює добутку кількості вузлів у дискретній моделі форми об'єкта на кількість степенів свободи.

Права частина системи лінійних алгебраїчних рівнянь визначається як вектор вузових сил (або значень, що відповідають впливу зовнішнього середовища на об'єкт). Вектор вузових сил так само, як і глобальна матриця, є спеціальною сумою локальних векторів. Його розмір відповідає розміру глобальної матриці.

Для забезпечення єдиності розв'язку отриманої системи лінійних алгебраїчних рівнянь необхідно враховувати крайові умови, задані для розв'язуваної задачі. Такими умовами, наприклад, виступають умови закріплення.

Значення шуканої величини у вузлах дискретної моделі (наприклад, переміщень) отримують у результаті розв'язання системи лінійних алгебраїчних рівнянь. Після чого на завершальному етапі виконують їх обробку: обчислюють похідні величини (наприклад, напруження), друкують звіти та візуалізують результати.

Отже, для скінченно-елементного аналізу можна сформулювати такий алгоритм:

algorithm fem

input:

M – дискретна модель форми об'єкта

P – множина сил, під дією яких знаходиться об'єкт

B – множина крайових умов

output:

X – значення шуканої величини у вузлах дискретної моделі

begin

$$[\mathbf{K}] \leftarrow \sum_{e \in \mathbf{M.E}} [\mathbf{k}_e]$$

$$\{\mathbf{F}\} \leftarrow \sum_{e \in \mathbf{M.E}} \{\mathbf{f}_e(\mathbf{P})\}$$

process_initial_values($[\mathbf{K}]$, $\{\mathbf{F}\}$, **B**)

```

 $\{\mathbf{X}\} \leftarrow [\mathbf{K}]^{-1} \{\mathbf{F}\}$ 
process_solution( $\{\mathbf{X}\}$ )
return  $\{\mathbf{X}\}$ 

```

При адаптивному скінченно-елементному аналізі обчислювальний процес є ітеративним. На кожній ітерації будуємо глобальні матрицю та вектор, враховуємо крайові умови та розв'язуємо систему лінійних алгебраїчних рівнянь. Далі за результатами обробки розв'язку або його похідних формуємо список елементів, які необхідно розбити. Виконуємо розбиття обраних елементів. Після чого починаємо наступну ітерацію. Алгоритм зупиняємо, якщо список елементів для розбиття був пустим (або виконано максимально допустиму кількість ітерацій розбиття). Алгоритм описаного методу можна подати у такому вигляді:

algorithm adaptive_fem

input:

R – функція, яка подає форму об'єкта Ω

M – дискретна модель форми об'єкта Ω

P – множина сил, під дією яких знаходиться об'єкт Ω

B – множина крайових умов

δ – обмеження на максимальне значення похибки на елементі

k – максимальна кількість ітерацій

ε – чисельний нуль

output:

M – адаптована дискретна модель форми об'єкта Ω

X – вузлові значення шуканої величини

begin

$i \leftarrow 1$

do

$[\mathbf{K}] \leftarrow \sum_{e \in M.E} [\mathbf{k}_e]$

$\{\mathbf{F}\} \leftarrow \sum_{e \in M.E} \{\mathbf{f}_e(\mathbf{P})\}$

process_initial_values($[\mathbf{K}]$, $\{\mathbf{F}\}$, B)

$\{\mathbf{X}\} \leftarrow [\mathbf{K}]^{-1} \{\mathbf{F}\}$

$C \leftarrow \text{character\_value}(\{\mathbf{X}\})$

$V \leftarrow \emptyset$

for each $e \in M.E$ **do**

if error(e , C) $> \delta$ **then**

$V \leftarrow V \cup M.\text{code}(e)$

end if

end for

subdivide(R , M , V , ε)

```

     $i \leftarrow i+1$ 
    while  $i \leq k$  and  $V \neq \emptyset$ 
    process_solution( $\{X\}$ )
    return  $\{X\}$ 

```

Як тестовий приклад розглянемо задачу дослідження напружено-деформованого стану фланця (рис. 3.8). В безрозмірних величинах товщину вважаємо рівною 20, модуль Юнга – 200000, коефіцієнт Пуассона – 0,33. Нехай на фланець діє сила, яка рівномірно розподілена вздовж отвору з радіусом 30, з інтенсивністю 20 (вектор сили направлений вздовж осі абсцис). Обидва отвори з радіусом 10 вважаємо жорстко закріпленими.

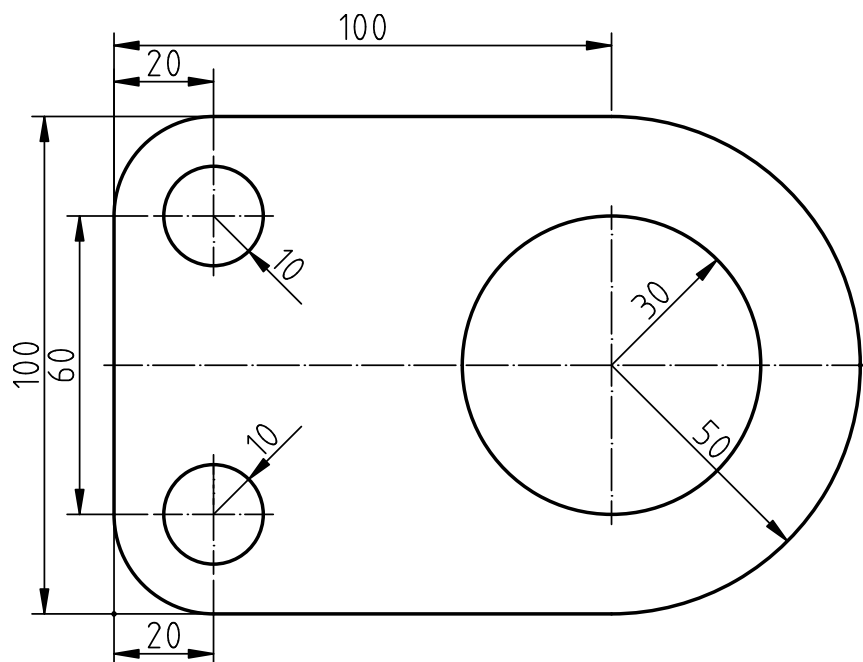


Рисунок 3.8 – Креслення фланця

Модель такого фланця можна описати з використанням функції

$$\text{flange}(x, y) = [\text{rectangle}(x-60, y-50, 80, 100) \vee \text{rectangle}(x-50, y-50, 100, 60) \vee \text{circle}(x-20, y-20, 20) \vee \text{circle}(x-20, y-80, 20) \vee \text{circle}(x-100, y-50, 50)] \wedge \neg \text{circle}(x-100, y-50, 30) \wedge \neg \text{circle}(x-20, y-20, 10) \wedge \neg \text{circle}(x-20, y-80, 10),$$

де circle – функція (1.11);

rectangle – функція (1.14).

Нехай вихідні дискретні моделі отримані за допомогою алгоритму background-grid-mesh з використанням рівномірної фонові сітки для 46×31 вузлів (рис. 3.9).

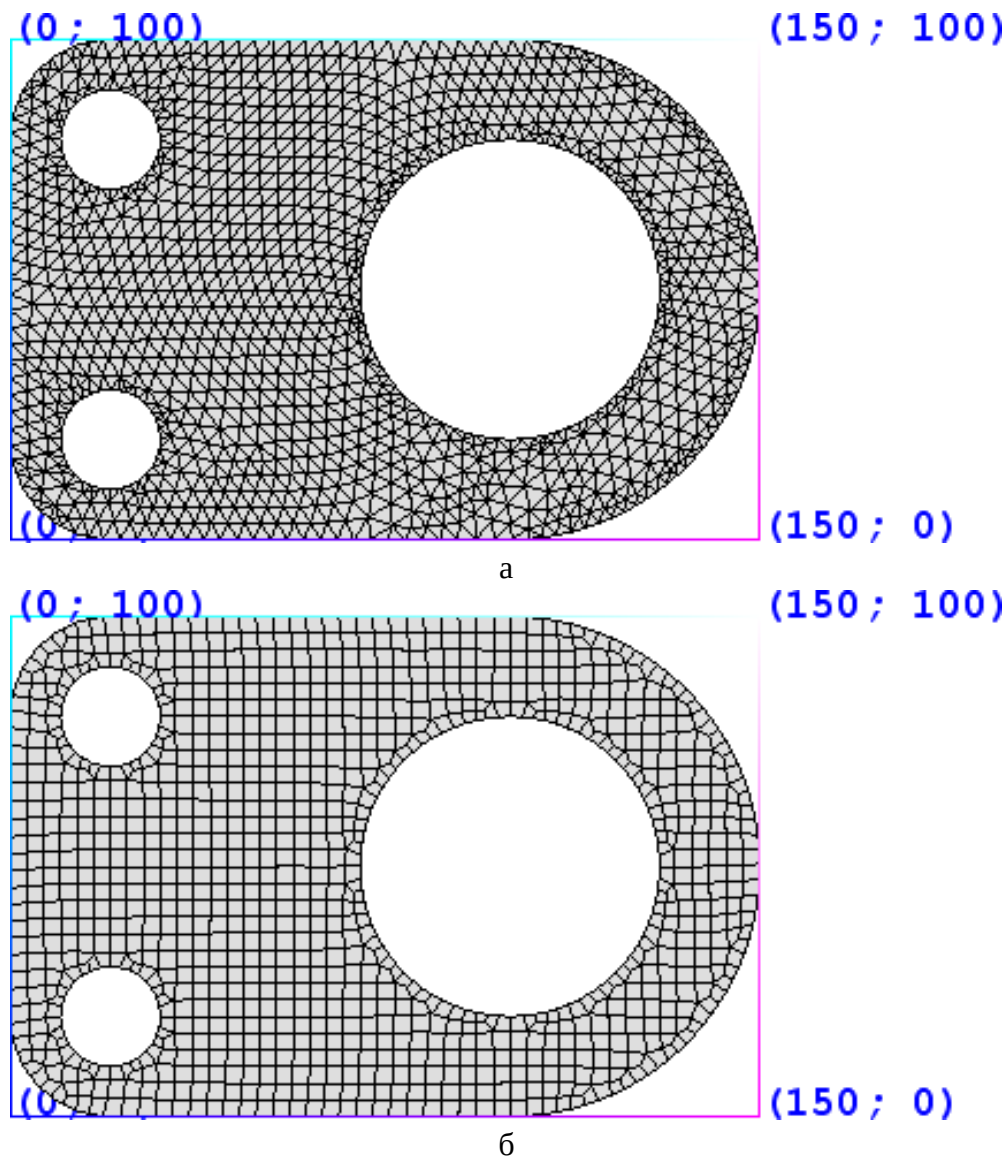


Рисунок 3.9 – Вихідна дискретна модель фланця: а – на базі трикутників; б – на базі чотирикутників

У результаті скінченно-елементного аналізу плоско-деформованого стану фланця отримаємо, якщо використати трикутні скінченні елементи (лінійні функції форми), то переміщення у першому напрямку (x) знаходяться в інтервалі $[0; 0,0497]$, у другому напрямі (y) – $[-0,0168; 0,0168]$, напруження: σ_{xx} – $[-24,683; 128,299]$, σ_{yy} – $[-67,017; 67,624]$, τ_{xy} – $[-43,924; 46,691]$, інтенсивність напружень за Мізесом σ_i – $[1,558; 132,909]$ (рис. 3.10).

Якщо використати чотирикутні скінченні елементи (білінійні функції форми), то переміщення у першому напрямі (x) знаходяться в інтервалі $[0; 0,0503]$, у другому напрямі (y) – $[-0,0172; 0,0172]$, напруження: σ_{xx} – $[-31,130; 141,285]$, σ_{yy} – $[-77,568; 71,964]$, τ_{xy} – $[-47,566; 47,151]$, інтенсивність напружень за Мізесом σ_i – $[2,343; 141,520]$ (рис. 3.10).

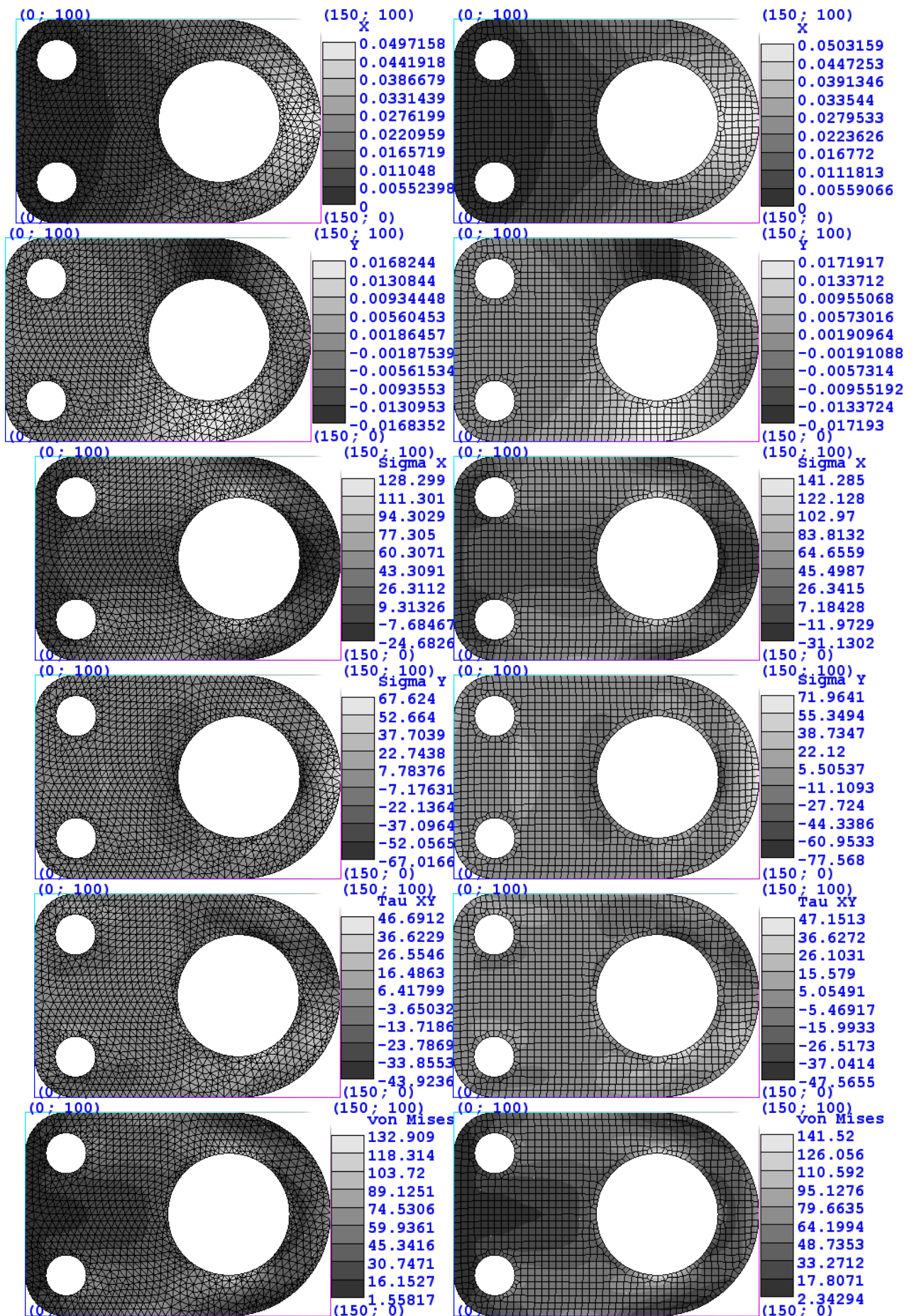


Рисунок 3.10 – Порівняння картин напружено-деформованого стану фланця при використанні трикутних і чотирикутних елементів

Поширені техніки знаходження областей для збільшення щільності вузлів і елементів використовують різні оцінки точності результату. Одним з найпростіших методів оцінки точності (якщо невідомий точний розв'язок) є порівняння зміни вузлових значень з середнім на елементі [334]. У такому випадку функція error, яка використана в алгоритмі адаптивного скінченно-елементного аналізу матиме такий вигляд:

$$\text{error}(e, \mathbf{C}) = \max_{i, j \in e} \frac{|\mathbf{C}_i - \mathbf{C}_j|}{\frac{1}{|e|} \left| \sum_{k \in e} \mathbf{C}_k \right|}. \quad (3.1)$$

Якщо використати формулу (3.1) в алгоритмі `adaptive_fem` з обмеженням максимальної похибки на елементі $\delta=0,4$ і максимальною кількістю ітерацій $k=3$ ($\mathbf{C}$ – довжина вектора переміщень), то для обох форм скінченних елементів щільність вузлів буде більшою біля отвору радіуса 10 (рис. 3.11).

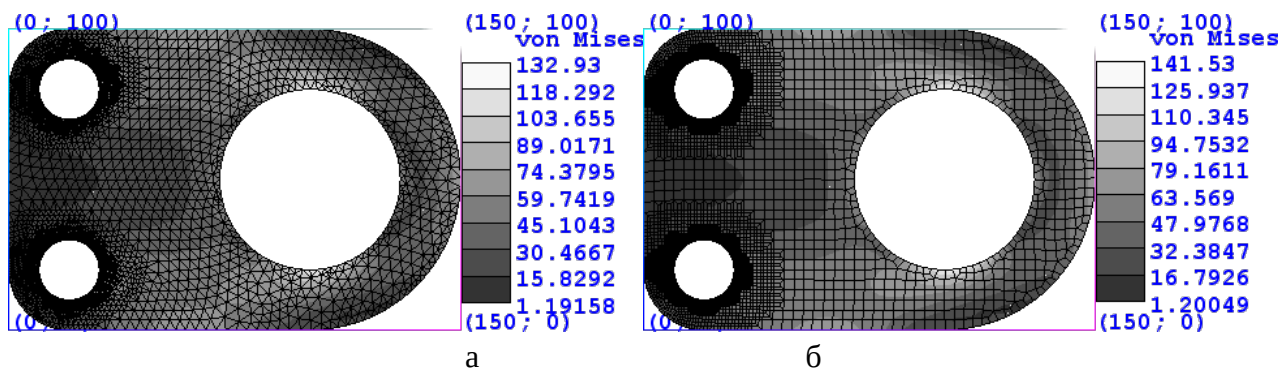


Рисунок 3.11 – Адаптивна дискретна модель фланця з використанням формули (3.1): а – на базі трикутників; б – на базі чотирикутників

Альтернативою формули (3.1) є градієнтна оцінка точності розв'язку, яка базується на ідеї збільшення щільності вузлів і елементів дискретних моделей форм об'єктів у областях з найбільшою зміною певного скаляра, що потребують особливої уваги для дослідника (наприклад, інтенсивність напружень при дослідженні міцності). Як результат функція error, яка використана в алгоритмі адаптивного скінченно-елементного аналізу матиме такий вигляд:

$$\text{error}(e, \mathbf{C}) = \max_{i, j \in e} \frac{\mathbf{C}_i - \mathbf{C}_j}{\max_k \mathbf{C}_k - \min_k \mathbf{C}_k}. \quad (3.2)$$

Якщо у алгоритмі `adaptive_fem` використати формули (3.2) з обмеженням максимально допустимої похибки на елементі $\delta=0,2$ і максимальною кількістю

ітерацій $k=3$ (C – інтенсивність напружень за Мізесом σ_i), то для обох форм скінченних елементів збільшення щільності вузлів буде біля отвору радіуса 30, де найбільше відносна зміна σ_i (рис. 3.12).

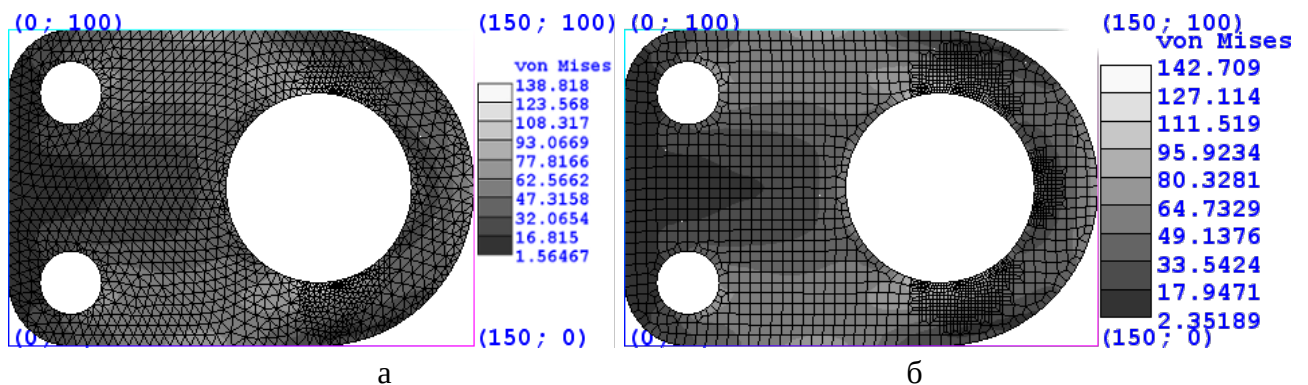


Рисунок 3.12 – Адаптивна дискретна модель фланця з використанням формули (3.2): а – на базі трикутників; б – на базі чотирикутників

Для дослідження напружено-деформованого стану фланця у тривимірні постановці можна використати таку функцію, що подає його форму:

$$\text{flange3}(x, y, z) = \text{flange}(x, y) \wedge z \wedge (20 - z).$$

У тривимірному випадку, якщо для адаптивного скінченно-елементного аналізу використовувати дискретні моделі на базі і тетраедрів (рис. 3.13, а) і шестигранників (рис. 3.13, б) з обмеженням максимальної похибки на елементі $\delta=0,2$ і максимальною кількістю ітерацій $k=3$ (C – інтенсивність напружень за Мізесом σ_i), то зони з найбільшою щільністю вузлів такі ж самі як у наведених вище двовимірних розв'язках (рис. 3.12).

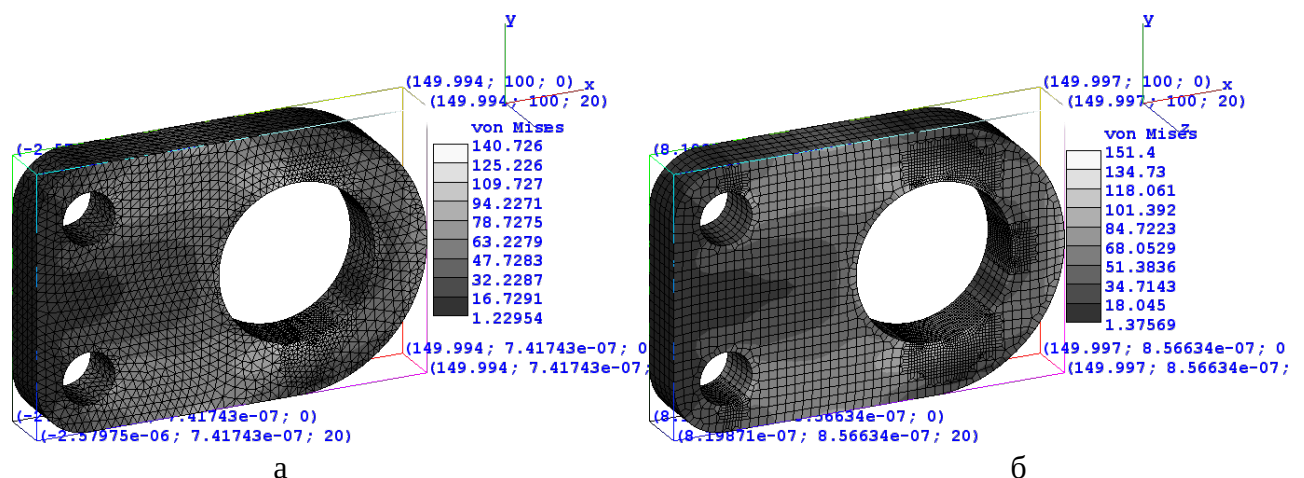


Рисунок 3.13 – Адаптивна тривимірна дискретна модель фланця з використанням формули (3.2): а – на базі тетраедрів; б – на базі шестигранників

Зазначимо, що для цього прикладу різниця чисельних значень для напружень між звичайним і адаптивним підходами складає декілька відсотків. Якщо порівняти результати плоско-напруженого стану з тривимірними, то різниця складає приблизно 10%, що обумовлено використаними гіпотезами для двовимірного випадку.

Для порівняння результатів використання адаптивних дискретних моделей форм об'єктів з відомими аналітичними розв'язками розглянемо задачу дослідження у тривимірній постановці напружено-деформованого стану стрижня, якій знаходиться під дією власної ваги (з параметрами як у пункті 2.1.2). Якщо використовувати вихідні рівномірні дискретні моделі, наведені на рис. 2.5, то для моделі на базі тетраедрів з обмеженням максимальної похибки на елементі $\delta=0,05$ і максимальною кількістю ітерацій $k=3$ (σ_z – напруження σ_z) напруження будуть σ_z у проміжку $[0,006; 1,53]$ МПа.

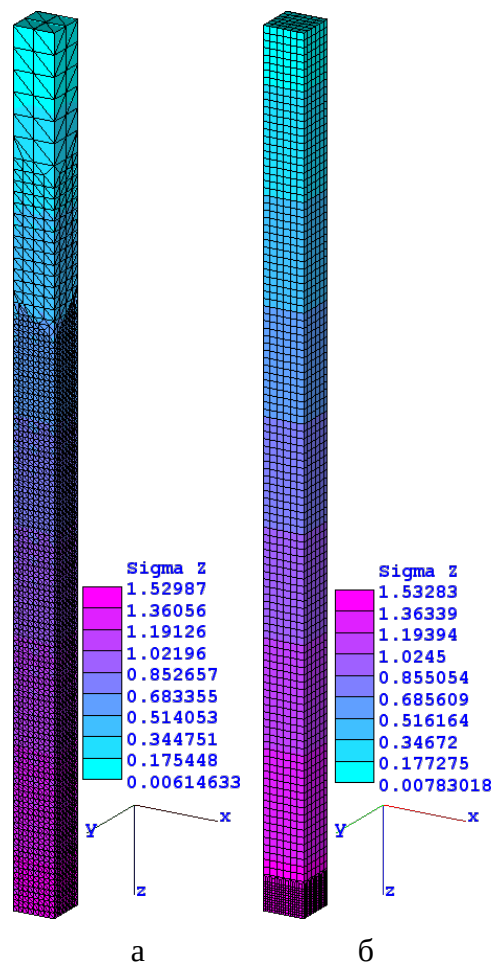


Рисунок 3.14 – Адаптивна тривимірна дискретна модель стрижня:
а – на базі тетраедрів; б – на базі шестигранників

Якщо використовувати модель на базі шестигранників з обмеженням максимальної похибки на елементі $\delta=0,02$ і максимальною кількістю ітерацій $k=3$, то напруження будуть σ_z в інтервалі $[0,009; 1,53]$ МПа. Аналітичний

розв'язок (2.8) відповідає інтервалу $[0; 1,53]$ МПа, для вихідної моделі на базі тетраедрів $\sigma_z \in [0,006; 1,29]$ МПа, для кубиків – $\sigma_z \in [0,022; 1,51]$ МПа. Отже, у обох випадках уточнюються екстремальні значення напружень σ_z . Водночас якісно дискретні моделі перебудовуються біля вільного кінця стрижня.

Як додатковий приклад застосування методу генерації адаптивних дискретних моделей розглянемо задачу дослідження поведінки балки з отворами малого радіуса (рис. 3.15). Нехай лівий край балки затиснений, а на правий діє сила $P=1000$ lbs. Довжина балки дорівнює 10 in. Два круглих отвори радіуса $R=0,045$ in знаходяться на верхній і нижній гранях на відстані 9 in.

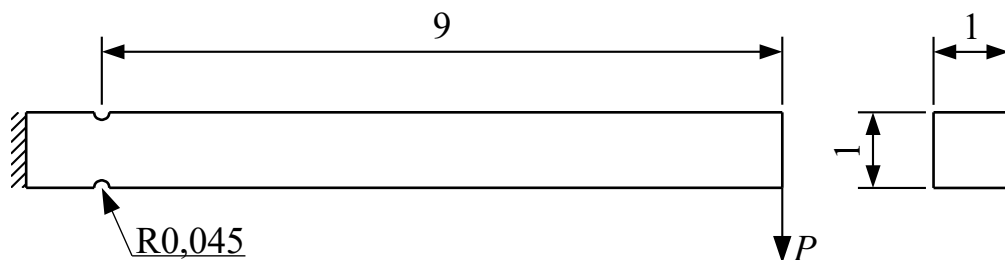


Рисунок 3.15 – Балка з отворами малого радіуса

У роботі [335] порівняно аналітичний і скінченно-елементний розв'язки на базі системи ANSYS для обчислення концентраторів напруження у горизонтальному напрямі при модулі Юнга $E=1$ psi і коефіцієнті Пуассона $\nu=0$. Так у цій роботі показано, що аналітична оцінка напружень у горизонтальному напрямі біля отворів складає 156849,3 psi. Водночас обчислення у системі ANSYS при використанні дискретної моделі на базі чотирикутників з 16 елементами для апроксимації сторони отвору максимальне значення напружень у горизонтальному напрямі дорівнює 124994 psi.

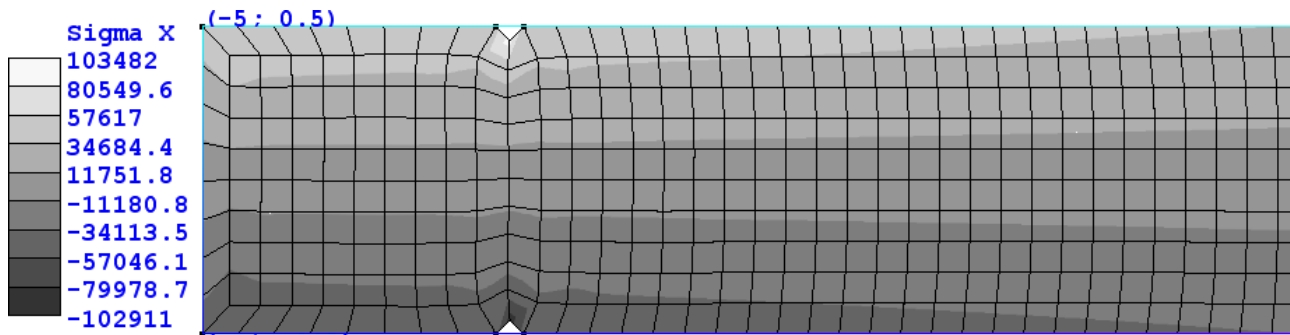
Функціональну модель форми такої балки можна подати функцією

$$\text{beam}(x, y) = \text{rectangle}(x, y, 10, 1) \wedge \\ \wedge \neg \text{circle}(x+4; y-0,5; 0,045) \wedge \neg \text{circle}(x+4; y+0,5; 0,045),$$

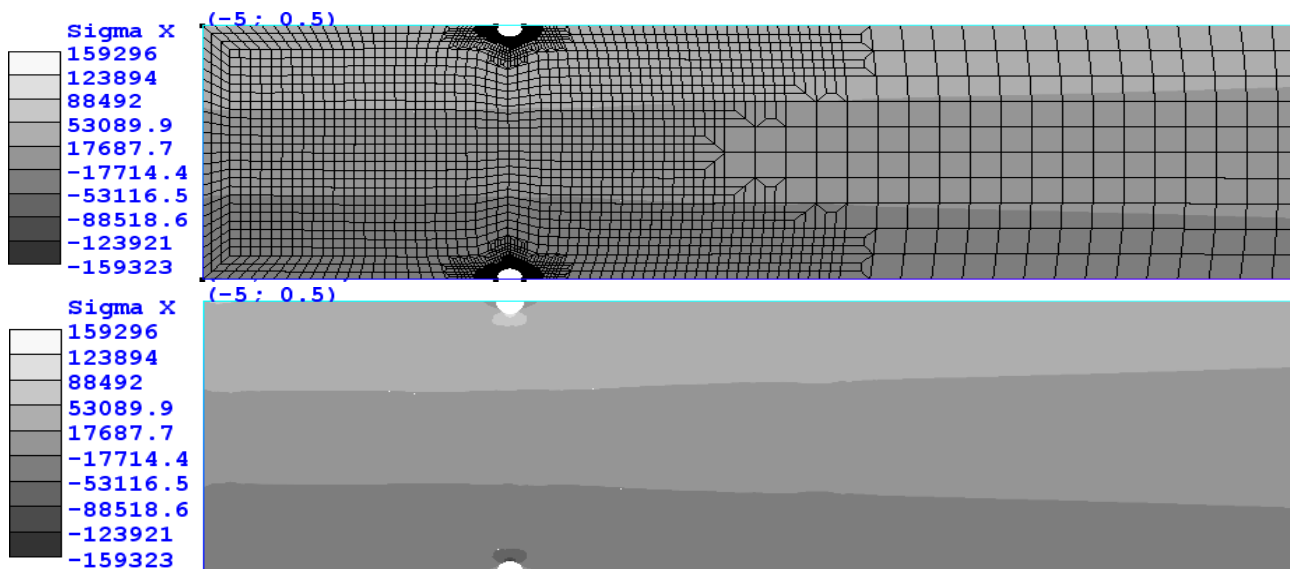
де circle – функція (1.11);

rectangle – функція (1.14).

Якщо як вигідну для узагальненого методу генерації дискретних моделей (алгоритм background-grid-mesh) використати рівномірну структуровану сітку чотирикутників на 101×11 вузлі, то отримаємо дискретну модель, яка має 1103 вузли та 996 елементів (рис. 3.16). За результатами скінченно-елементного аналізу на базі цієї моделі максимальне значення напружень σ_x дорівнює 103482 psi.

Рисунок 3.16 – Напруження σ_x у балці

Якщо використати адаптивний скінченно-елементний аналіз із обмеженням максимальної похибки на елементі $\delta=0,1$ і максимальною кількістю ітерацій $k=4$ (С – напруження σ_x), то максимальне за модулем значення напружень у першому напрямі дорівнюватиме 159323 psi. Водночас у підсумковій дискретній моделі буде вже 4975 вузлів і 4744 елементи, щільність яких буде більшою біля отворів (рис. 3.17).

Рисунок 3.17 – Напруження σ_x у балці (адаптивна дискретна модель)

Отже, використання адаптивної генерації дискретних моделей форм об'єктів дозволяє підвищити точність математичного моделювання. Проте такий підхід є екстенсивним щодо використання обчислювальних ресурсів.

3.3. Візуалізація чисельних даних на поверхні конструкції

Результатом скінченно-елементного аналізу поведінки конструкції є масиви чисельних даних. Прикладами таких масивів є вузлові значення

переміщень, деформацій або напружень. Такі масиви потрібно подавати у доступній для дослідника/проектувальника формі. Водночас виникають такі основні задачі:

- перевірка достовірності, точності й адекватності результатів до фізичної постановки задачі;
- синтез допоміжної інформації (наприклад, екстремальних значень числових масивів, зон найбільших градієнтів тощо).

Сучасні функціональні САПР, що використовують метод скінченних елементів для інженерного аналізу, мають спеціальні модулі або підсистеми, які призначені для розв'язання зазначених задач. Результатом роботи таких модулів або підсистем, зазвичай, є візуальний образ масивів числових значень з допоміжною звітною інформацією.

Найчастіше числові масиви візуалізують дво- або тривимірними графіками, картинами ліній рівня або векторними полями. Числові масиви, розподілені по деякій дво- або тривимірній області, подають напівтоновими або кольоровими картинами, в яких кожен колір відповідає певному діапазону значень.

Зазвичай видима частина дискретної моделі форми конструкції є сукупністю плоских фігур простої форми. У тривимірному випадку видимими є грані елементів – трикутники або чотирикутники. Отже, задача візуалізації скінченно-елементної моделі та обчисленого на її базі числового масиву може бути замінена побудовою зафарбованих за певним правилом видимих граней.

Математичний апарат, призначений для відображення геометричних фігур на екран та видалення невидимих ліній, вельми ефективно реалізовано засобами таких графічних бібліотек як DirectX і OpenGL. У хмарних середовищах візуалізацію числових масивів можна виконувати на сторонах клієнта та сервера. У випадку візуалізації на стороні клієнта можна використовувати такі бібліотеки як, наприклад, Three.js⁵ або Babylon.js⁶. На стороні сервера можна фільтрувати числові масиви для передачі клієнтам лише елементів, що знаходяться “у кадрі” (потрапляють у налаштовану клієнтом область перегляду). Альтернативно на стороні сервера можливо генерувати двовимірну проекцію моделі та пересилати її клієнту як двовимірну картинку.

Якщо врахувати, що найбільш поширені форми граней можна об'єднанням трикутників (наприклад, чотирикутні грані можна подавати парами трикутників проведенням однієї з діагоналей), то достатньо розглянути спосіб візуалізації трикутних граней. Можливим розв'язанням задачі побудови картини розподілу значень числового масиву у трикутнику є такий підхід.

5 <https://threejs.org/>

6 <https://www.babylonjs.com/>

Нехай деякий числовий масив $\mathbf{A}$ містить дійсні значення з відрізка $[a_{\min}; a_{\max}]$, а також задана кількість кольорових зон n . У такому випадку відрізок значень $[a_{\min} + ih; a_{\min} + (i+1)h]$, де $h = \frac{a_{\max} - a_{\min}}{n-1}$, $i = \overline{0, n-1}$, буде відповідати кольору з номером i . Отже, номер кольору, що відповідає значенню $a \in [a_{\min}; a_{\max}]$, можна визначити за допомогою такої функції:

$$\text{color}(a) = \left\lfloor \frac{a - a_{\min}}{h} \right\rfloor,$$

де $\lfloor x \rfloor$ – операція знаходження найбільшого цілого, меншого, ніж x .

Якщо трикутник, заданий вузлами з радіус-векторами $\mathbf{A}$, $\mathbf{B}$ і $\mathbf{C}$, яким відповідають числові значення a_A , a_B і a_C , то можна використовувати такий алгоритм.

algorithm triangle_color

input:

$\mathbf{A}$, $\mathbf{B}$ і $\mathbf{C}$ – координати вузлів трикутника

a_A , a_B і a_C – вузлові значення

output:

$L = \{(t = (\mathbf{A}_t, \mathbf{B}_t, \mathbf{C}_t), \text{clr}_t)\}$ – множина пар: трикутник-колір

begin

$L \leftarrow \emptyset$

if $\text{color}(a_A) = \text{color}(a_B)$ **and** $\text{color}(a_A) = \text{color}(a_C)$ **then**

$L \leftarrow \{((\mathbf{A}, \mathbf{B}, \mathbf{C}), \text{color}(a_A))\}$

else

$L_A \leftarrow \text{triangle\_color}\left(\mathbf{A}, \frac{\mathbf{A} + \mathbf{B}}{2}, \frac{\mathbf{A} + \mathbf{C}}{2}, a_A, \frac{a_A + a_B}{2}, \frac{a_A + a_C}{2}\right)$

$L_B \leftarrow \text{triangle\_color}\left(\mathbf{B}, \frac{\mathbf{B} + \mathbf{C}}{2}, \frac{\mathbf{B} + \mathbf{A}}{2}, a_B, \frac{a_B + a_C}{2}, \frac{a_B + a_A}{2}\right)$

$L_C \leftarrow \text{triangle\_color}\left(\mathbf{C}, \frac{\mathbf{C} + \mathbf{A}}{2}, \frac{\mathbf{C} + \mathbf{B}}{2}, a_C, \frac{a_C + a_A}{2}, \frac{a_C + a_B}{2}\right)$

$L_{ABC} \leftarrow \text{triangle\_color}\left(\frac{\mathbf{A} + \mathbf{B}}{2}, \frac{\mathbf{B} + \mathbf{C}}{2}, \frac{\mathbf{A} + \mathbf{C}}{2}, \frac{a_A + a_B}{2}, \frac{a_B + a_C}{2}, \frac{a_A + a_C}{2}\right)$

$L \leftarrow L_A \cup L_B \cup L_C \cup L_{ABC}$

end if

return L

Приклади візуалізації розподілу значень у трикутнику з використанням різних кольорових схем наведено на рис.

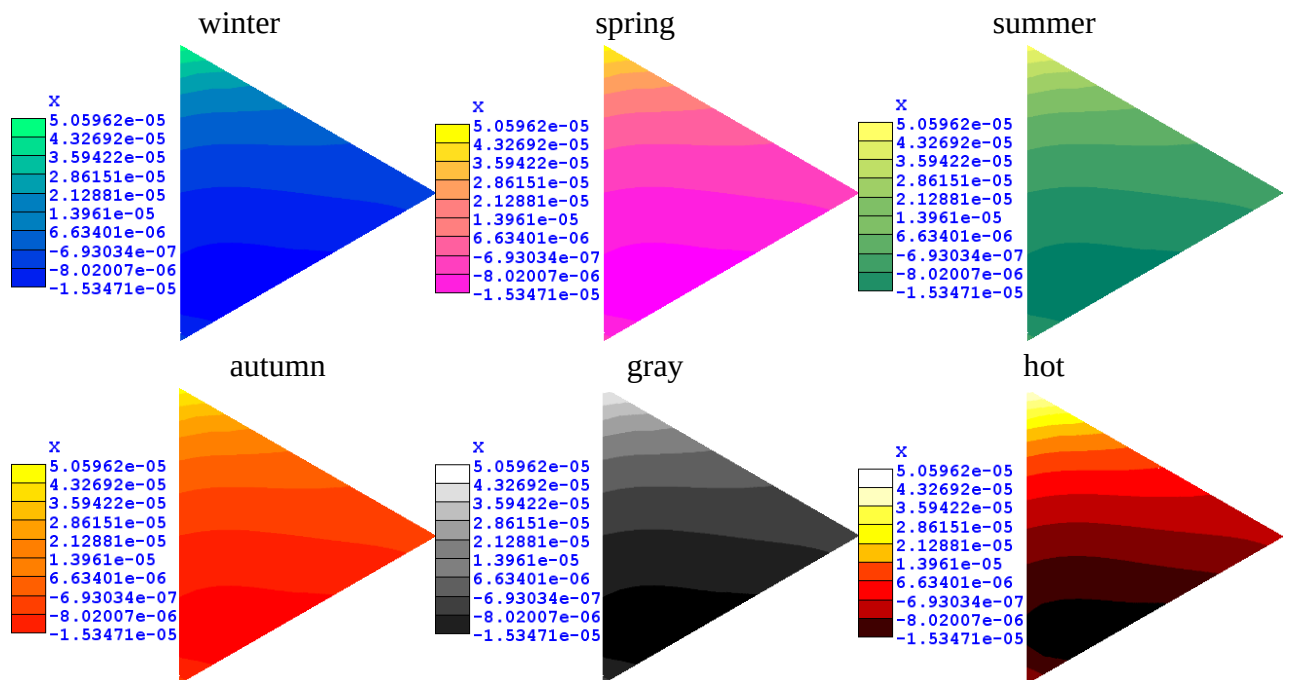


Рисунок 3.18 – Приклади кольорових схем для візуалізації розподілу числових значень на трикутнику

Необхідно зазначити, що на більшості ілюстрацій у монографії використано одну з цих схем для візуалізації розподілів переміщень або напружень, а також значень R-функцій.

РОЗДІЛ 4. КАРКАС ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Завдання, з якими зустрічається сучасне програмне забезпечення, все більше ускладнюються. В основу обчислювальних модулів закладають різні підходи та методи. Як результат суттєво ускладнюються розробка та підтримка. Можливим способом спрощення розробки та підтримки такого програмного забезпечення є не тільки розробка універсальних узагальнених методів, а й типових програмних рішень, які можуть використовуватися повторно.

4.1. Шаблони проектування для об'єктно-орієнтованих САПР машинобудування

Шаблони проектування, як загальні багаторазово використовувані рішення, вперше були розроблені в кінці 1980х років і з того часу їх активно використовують в інженерії програмного забезпечення [301–304]. До недавнього часу розробники наукомісткого програмного забезпечення зазвичай уникали об'єктно-орієнтованого підходу через його накладні обчислювальні витрати [302]. Проте, програмне забезпечення САПР машинобудування стає більшим і складнішим. Виникають питання щодо його гнучкості, розширюваності та супроводу [336]. Шаблони програмного забезпечення розв'язують ці задачі шляхом надання загальних об'єктно-орієнтованих рішень.

Архітектори, механіки, винахідники та інженери для розв'язання інженерних задач використовують підсистеми інженерного аналізу (Computer-Aided Engineering, CAE) САПР машинобудування. Підсистеми інженерного аналізу є наукомісткими частинами САПР, які можуть включати модулі дослідження напружено-деформованого стану, поширення температури, динаміки твердих тіл, газів і рідин. Загалом, підсистема інженерного аналізу має три підсистеми: препроцесор, процесор і постпроцесор результатів.

Препроцесор відповідає за визначення моделі задачі. Зазвичай модель задачі включає опис форми об'єкта, впливів зовнішнього середовища (наприклад, прикладених сил, температур, вібрацій). Форму досліджуваного об'єкта описують за допомогою однієї з схем подання. Під схемою подання розуміють семантично та синтаксично коректне відношення між множинами формальних описів і допустимих форм об'єктів.

Процесор є ядром підсистеми інженерного аналізу. Процесор – це множина чисельних методів, які зазвичай реалізуються з використанням високопродуктивних обчислень.

Можливо, найбільш поширеним чисельним методом є метод скінченних елементів. У цьому методі неперервна модель замінюється дискретною. Відповідно, перший крок такого моделювання потребує дискретизації неперервної геометричної області на елементи простої форми, наприклад, трикутники, чотирикутники, тетраедри та шестигранники.

Поширені альтернативи методу скінченних елементів (наприклад, безсіткові методи) можуть так само бути використані процесором. Ці методи не вимагають зв'язків між вузлами у моделі форми об'єкта. Проте, вузлова дискретна модель все ще необхідна. Отже, генерація дискретних моделей є важливою складовою, яка з'єднує опис форми об'єкта з його дискретною моделлю.

Після застосування чисельного аналізу на основі значень, отриманих у вузлах дискретної моделі, постпроцесор обчислює похідні величини та візуалізує результати. Тобто, постпроцесор створює подання дискретної моделі. Також значення, отримані у вузлах дискретної моделі, можна використовувати для поліпшення обох вихідної (неперервної) та проміжної (дискретної) моделей.

Отже, абстракціями верхнього рівня при розробці підсистем інженерного аналізу є препроцесор, процесор і постпроцесор. Розробка шаблонів проектування дозволить об'єднати ці абстракції в єдину об'єктно-орієнтовану модель.

4.1.1 Декомпозиція та ідентифікація

Зазвичай, підсистеми інженерного аналізу САПР машинобудування та їх базис чисельних методів розробляють з використанням процедурного підходу та таких мов програмування як FORTRAN, C тощо. Процедурний підхід вельми ефективний у реалізації чисельних методів, включаючи метод скінченних елементів. Проте, підтримка САПР машинобудування (як наукомісткого програмного забезпечення) включає оновлення чисельних моделей, додавання нової функціональності та підвищення продуктивності програмного кода.

Розглянемо підсистему САПР машинобудування з процесором, що ґрунтується на методі скінченних елементів. Можна припустити, що модель кожного об'єкта спочатку описана за допомогою однієї зі схем подання, а також побудована відповідна дискретна модель його форми.

Виконаємо декомпозицію САПР машинобудування з використанням аналізу загальностей та частковостей (commonality and variability analysis) [337].

Результати ідентифікації абстракцій (загальностей) та конкретних реалізацій (частковостей) наведені у табл. 4.1.

Таблиця 4.1 – Абстракції та реалізації САПР машинобудування

Абстракція	Конкретні реалізації
Схема подання (Representation scheme)	граничне подання (BRep); конструктивна блокова геометрія (CSG); функціональне подання (FRep) тощо;
Точка (Point)	двовимірна точка; тривимірна точка;
Вузол (Node)	двовимірний вузол; тривимірний вузол;
Елемент (Element)	сегмент; трикутник; чотирикутник; тетраедр; шестигранник;
Дискретна модель (Mesh)	двовимірна дискретна модель; тривимірна дискретна модель;
Двовимірна дискретна модель	двовимірна дискретна модель на базі трикутників; двовимірна модель на базі чотирикутників; двовимірна модель на базі сегментів;
Тривимірна дискретна модель	дискретна модель на базі тетраедрів; дискретна модель на базі шестигранників; тривимірна дискретна модель на базі трикутників; тривимірна дискретна модель на базі чотирикутників;
Процесор МСЕ	процесор двовимірних задач; процесор тривимірних задач;
Процесор двовимірних задач	процесор двовимірних задач пружності; процесор задач дослідження пластин;
Процесор тривимірних задач	процесор тривимірних задач пружності; процесор дослідження оболонок;

4.1.2 Каталог шаблонів проектування

4.1.2.1 Шаблон проектування «UI–Model–Analysis»

Сучасні програмні підсистеми інженерного аналізу САПР машинобудування мають інтегрований графічний інтерфейс користувача (UI).

Використовуючи контролери графічного інтерфейсу користувач вводить модель задачі (Model) та ініціює її аналіз процесором (Analysis). Отже, на верхньому рівні абстракції три основних пакета беруть участь у базовому шаблоні «UI–Model–Analysis» (рис. 4.1). Метою цього шаблону проектування є декомпозиція підсистеми інженерного аналізу на високорівневі частини, а також відділення класів, які пов'язані з графічним інтерфейсом від класів моделі та процесора.

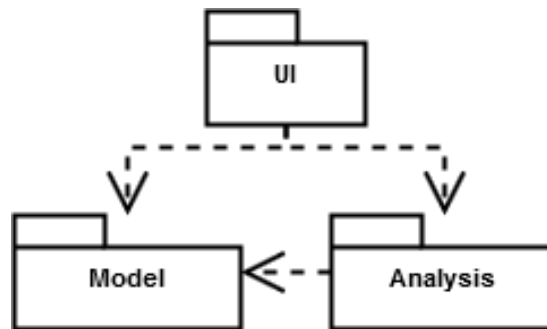


Рисунок 4.1 – Пакети шаблону проектування «UI–Model–Analysis»

Пакет UI (графічний інтерфейс) містить класи, які реалізують пов'язані з графічним інтерфейсом функції програмного забезпечення. Ці класи об'єднують функції пре- і постобробки (наприклад, вводу даних і візуалізацію). Отже, не повинно бути зв'язності між підсистемами аналізу та графічного інтерфейсу. Проте, пакет UI залежить від пакетів Model (препроцесор) і Analysis (процесор).

Загалом модель визначають за допомогою деякої проблемно-орієнтованої мови. Проблемно-орієнтована мова дозволяє описати форму об'єкта та параметри задачі (фізико-механічні характеристики, впливи зовнішнього середовища). Форму об'єкта описують на базі певної схеми подання. Абстрактна схема подання (Representation) є інтерфейсом, який дозволяє перевірити належність точки до моделі об'єкта. Отже, схема подання є однією з ключових абстракцій пакета Model.

Пакет Analysis (процесор) відповідає за чисельний аналіз. Процедури аналізу, зазвичай, виконують за допомогою метода скінченних елементів. У цьому методі два основних етапи: генерація дискретної моделі та безпосередньо скінченно-елементний аналіз. Отже, дискретна модель (Mesh) і процесор задач метода скінченних елементів (FEA Problem) є ключовими абстракціями цього пакета.

4.1.2.2 Шаблон проектування «Representation–Mesh»

Припустимо, що форма об'єкта описана у термінах деякої схеми подання. Найбільш поширеними схемами подання є граничне подання (BRep),

конструктивна блокова геометрія (CSG) і функціональне подання (FRep). Спільною властивістю конструктивної блокової геометрії та функціонального подання є можливість легкої перевірки приналежності точки до моделі об'єкта. Водночас (як це показано у розділі 2) граничне подання загалом потребує перевірки парності перетинів променя з точки і межі моделі об'єкта. Тим не менш, спільною поведінкою відповідних класів можна вважати класифікацію точок щодо межі об'єкта.

Для генерації дискретної моделі застосовуються певні методи розбиття частини евклідового простору на елементи простої форми. Можна припустити, що дискретна модель – це абстрактний інтерфейс (Mesh), який дозволяє будувати та ітерувати через колекцію елементів. У такому випадку конкретні класи будуть похідними від абстрактного класу Mesh (рис. 4.2). Конкретні класи генерують елементи певної форми, використовуючи інтерфейс абстракції схеми подання (Representation) для класифікації точок простору. Обидва класи Representation і Mesh беруть участь у шаблоні проектування «Representation–Mesh». Метою цього шаблону проектування є розподіл обов'язків між класами подання форм об'єктів і генерації дискретних моделей.

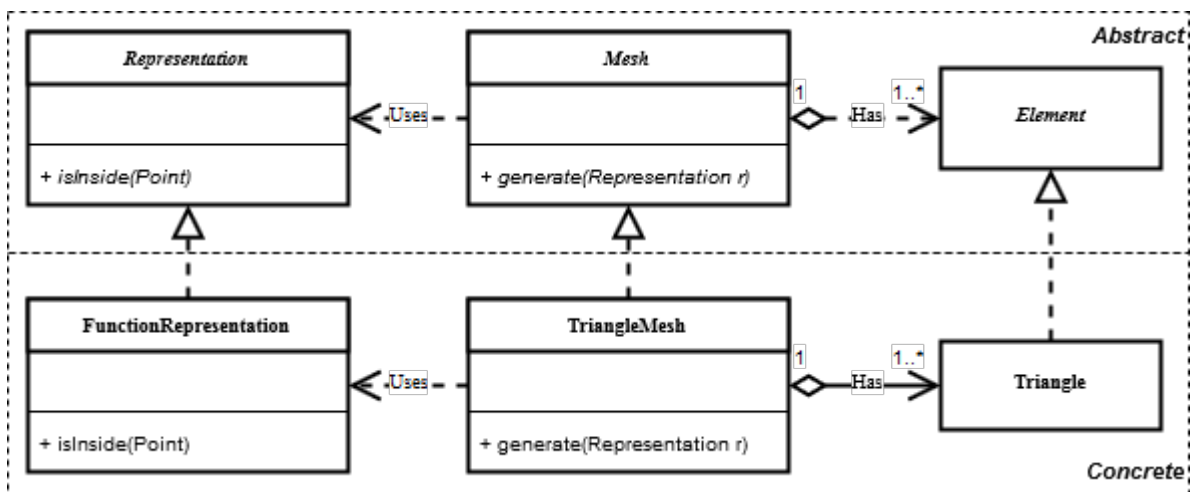


Рисунок 4.2 – Шаблон проектування «Representation–Mesh»

Обидва класи Representation і Mesh можуть бути окремими випадками шаблону проектування «Стратегія» (Strategy [303]). Клас Mesh та його похідні можна реалізувати як шаблон проектування «Ітератор» (Iterator [303]) для організації перегляду колекцій вузлів і елементів. Водночас деякі методи генерації дискретних моделей форм об'єктів (наприклад, розглянуті у розділах 4 і 5 узагальнені методи) можна реалізувати як «Шаблонний метод» (Template Method [303]).

4.1.2.3 Шаблон проектування «Element–Node»

Загалом елемент дискретної моделі (Element) – це впорядкована колекція вузлів (Node). У обох (двовимірному та тривимірному) випадках елементи мають ребра. Ребро (Edge) – це відрізок прямої, що з'єднує два вузли. Тривимірні елементи на додаток до ребер мають грані. Грань (Face) – це плоский елемент, обмежений ребрами. Тобто, конкретна грань – це об'єкт класу, який реалізує інтерфейс Element. Отже, Element, Face, Edge і Node – структурні частини шаблону проектування «Element–Node» (рис. 4.3). Метою цього шаблону проектування є розробка об'єктної декомпозиції для визначення елементів.

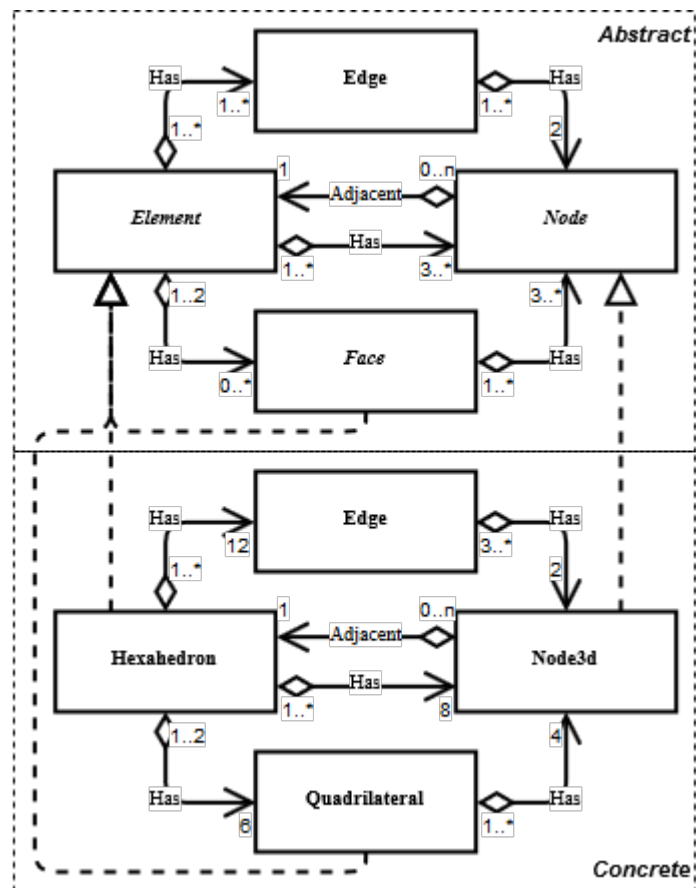


Рисунок 4.3 – Шаблон проектування «Element–Node»

Клас Node – це абстрактний інтерфейс, який визначає спільну для всіх типів вузлів поведінку (наприклад, маніпуляція типами вузлів, управління множиною інцидентних елементів тощо). Можуть бути різні реалізації для конкретних класів вузлів. Наприклад, операції зберігання координат і геометричні операції над ними можна реалізувати безпосередньо у класі. Альтернативно, клас вузла може бути композицією об'єктів. У такому випадку

будуть отримані паралельні ієрархії для класів, що реалізують операції над вузлами і векторами.

Клас *Element* і його похідні (включно грані) можна реалізувати з використанням шаблону проектування «Ітератор» для організації проходу по колекціям вузлів, ребер і граней. Шаблон «Ітератор» також можна використати для реалізації роботи зі списками інцидентних елементів.

4.1.2.4 Шаблон проектування «FEA Problem»

Спочатку метод скінченних елементів був розроблений для дослідження напружено-деформованого стану, а потім застосований у розв'язанні інших задач, що ґрунтуються на рівняннях у частинних похідних. У інженерних задачах, зазвичай, необхідно визначити розподіл значень деякої невідомої величини (наприклад, переміщень у задачах дослідження напружено-деформованого стану). У методі скінченних елементів розподіл значень невідомої величини апроксимують кусково-лінійними функціями на кожному елементі. Як результат отримують систему лінійних алгебраїчних рівнянь, яка дозволяє обчислити значення невідомої величини.

З математичної точки зору кожен елемент можна подати матрицями та векторами. Операції ансамблювання дозволяють трансформувати значення з локальних матриць і векторів у глобальні. Вихідною інформацією у процесі анасамблювання є номери вузлів і зв'язки між ними.

Перед розв'язанням системи лінійних алгебраїчних рівнянь повинні бути враховані крайові умови. Вони спрощують глобальні матрицю і вектор шляхом виключення деяких невідомих. Якщо крайова умова застосована до деякого вузла, то виключаються рядок і стовпець глобальної матриці, які визначені глобальним номером вузла і напрямком крайової умови.

Локальні та глобальні вектори обчислюються при врахуванні сил, що діють в вузлах елементів. Аналогічно крайовим умовам, конкретна позиція вузлового навантаження обчислюється на базі глобального номера вузла та напрямку, в якому це навантаження діє.

Шаблон проектування «FEA Problem» можна отримати з шаблону «Шаблонний метод» [303] шляхом додавання дискретної моделі, крайових умов і сил (рис. 4.4). Метою цього шаблону проектування є визначення структури алгоритму скінченно-елементного аналізу та об'єктної композиції для крайових умов та сил, які беруть участь в задачі.

Сили та крайові умови реалізують інтерфейс *FeaValue*. Цей інтерфейс дозволяє отримати напрям дії та значення в довільній точці. Крайові умови та сили зазвичай визначаються користувачем з використанням графічного інтерфейсу або проблемно-орієнтованої мови. Проте, інтерфейс *FeaValue* можна реалізувати за допомогою шаблону проектування «Адаптер» (Adapter [303]).

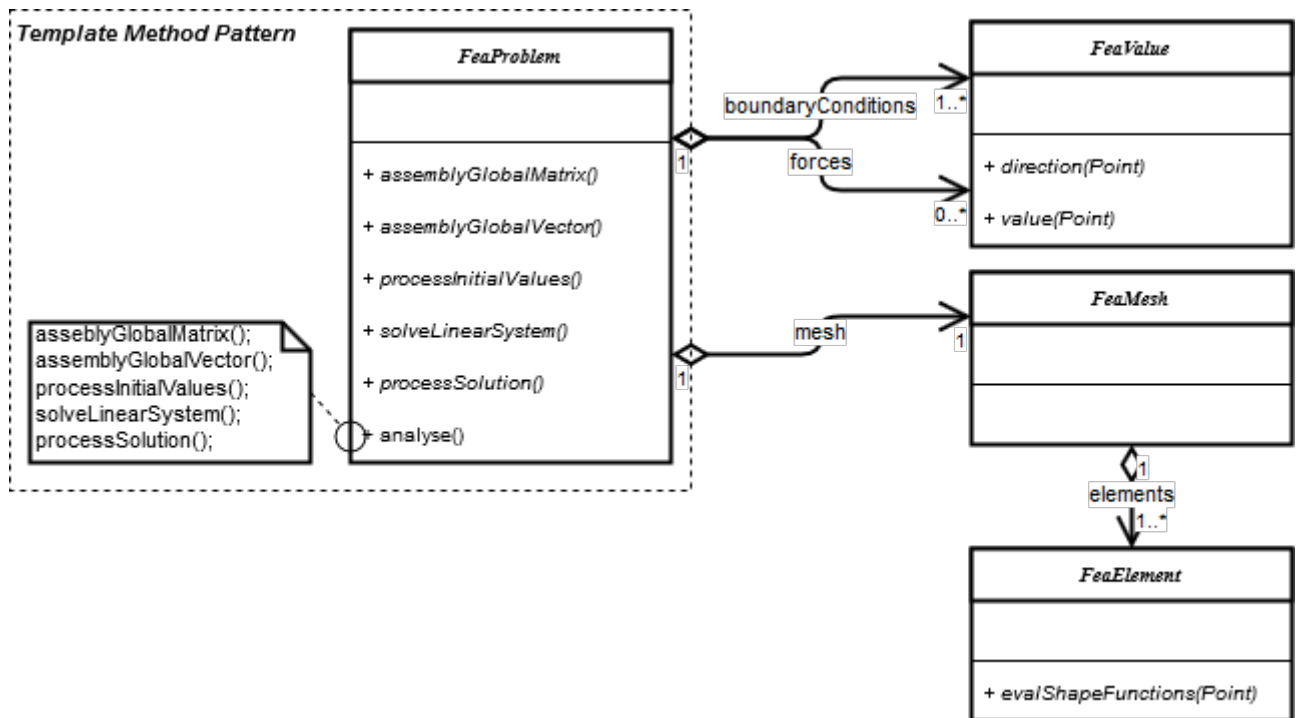


Рисунок 4.4 – Шаблон проектування «FEA Problem»

Важливим кроком процесу ансамблювання є обчислення інтерполяційних функцій (функцій форми) для кожного елемента. Зазвичай як функції форми використовують поліноми. Степінь полінома залежить від кількості вузлів у елементі. Найбільш поширеними є лінійні, білінійні та трилінійні. Поліноми більш високих степенів можна отримати додаванням додаткових вузлів у елементи. Отже, класи *FeaMesh* і *FeaElement*, основним обов'язком яких є обчислення функції форми, можна отримати із застосуванням шаблону проектування «Адаптер» до відповідних класів, що описані вище.

Розроблені вище шаблони основних логічних елементів програмної підсистеми інженерного аналізу САПР машинобудування пропонують схеми об'єктної декомпозиції та розподілу обов'язків без обмежень на реалізацію. Розподіл обов'язків між відносно незалежними блоками дозволяє спростити колективну розробку та підтримку САПР як складного програмного продукту.

4.2. Методи генерації та обробки дискретних моделей форм об'єктів на базі технологій паралельних обчислень

Паралельні обчислення є однією з базових комп'ютерних концепцій. Ідея одночасного використання декількох паралельних процесів для вирішення однієї задачі подібна до ідеї колективної праці для досягнення спільної мети. Паралельні обчислення відносяться до засобів зменшення часу, необхідного на виконання завдання. Отже, актуальною є розробка методів, що використовують паралельні обчислення.

4.2.1 Метод генерації дискретних моделей форм об'єктів з використанням паралельних обчислень

Для складних форм об'єктів генерація дискретних моделей може займати досить тривалий час. Можливим способом зменшення затраченого часу є залучення технологій паралельних обчислень.

Нехай у комп'ютерній системі є k функціональних пристроїв (процесорів, обчислювальних ядер тощо). У такому випадку фонову сітку, яка використовується як вихідна в узагальненому методі генерації дискретних моделей форм об'єктів, можна обробляти паралельно доступними пристроями. Для цього розіб'ємо фонову сітку на k підмножин. Таке розбиття можна здійснювати різними способами: наприклад, розбивши область на m горизонтальних або вертикальних смуг, або рекурсивно розбивши на k квадрантів (загалом, розбиття може залежати від щільності елементів фонові сітки). Потім до кожної підмножини можна одночасно застосувати дії, описані в узагальненому методі генерації дискретних моделей форм об'єктів. Критичними секціями будуть додавання нових вузлів і елементів у дискретну модель. Отже, алгоритм паралельної генерації дискретної моделі набуде вигляду наступної послідовності кроків.

1. Генерація фонові дискретної моделі $\mathbf{M}$ і розбиття її на k підмножин $\mathbf{M}_i$, $i \in 1..k$.

2. Паралельно: для кожної з підмножин $\mathbf{M}_i$, ($i \in 1..k$) видалення зовнішніх вузлів і елементів.

3. Послідовно: синхронізація підмножин для створення початкової дискретної моделі $\mathbf{M} = \mathbf{M}_1 \cup \mathbf{M}_2 \dots \mathbf{M}_k$ і видалення конфігурацій елементів, які можуть обумовлювати неоднозначність.

4. Послідовно: розбиття множини вузлів N дискретної моделі M на k рівних за розміром підмножин N_i , $i \in 1..k$ ($M.N = N_1 \cup N_2 \cup \dots \cup N_k$; $\forall i \neq j N_i \cap N_j = \emptyset$).

5. Паралельно: для кожного вузла, який належить N_i знайти перетин з границею моделі форми об'єкта.

6. Паралельно: для кожного $\delta(M_i)$ сформувати межовий шар елементів.

Запропонований алгоритм реалізований у вигляді програми на мові C++ з використанням бібліотеки OpenMP для організації паралельних обчислень. Як тестова платформа використаний комп'ютер з процесором Intel Core i5-3330 (3.0GHz, 4 ядра) і 8 гігабайтами оперативної пам'яті, який знаходиться під керуванням операційною системою LinuxMint 17.1 (64 біти, ядро Linux 3.13, компілятор gcc 4.8.2). У тестових прикладах використано рівномірні фонові дискретні моделі на базі трикутних елементів.

Графіки залежності часу генерації дискретної моделі планки, яка задана формулою (1.22), від кількості вузлів у дискретній моделі наведені на рис. 4.5. Як видно з рис. 4.5, якщо у фоновій дискретній моделі міститься 2000 вузлів, то час паралельної генерації з використанням двох потоків практично збігається з послідовним, а час, що використаний чотирма потоками, удвічі більший. Починаючи з 10000 вузлів генерації дискретної моделі з використанням чотирьох паралельних потоків стає швидшим за послідовну генерацію, а починаючи з 16000 – за генерацію двома паралельними потоками. З подальшим збільшенням кількості вузлів різниця у витраченому часі прагне до теоретичної оцінки. Ці результати обумовлені існуванням накладних обчислювальних витрат на створення та управління паралельними потоками.

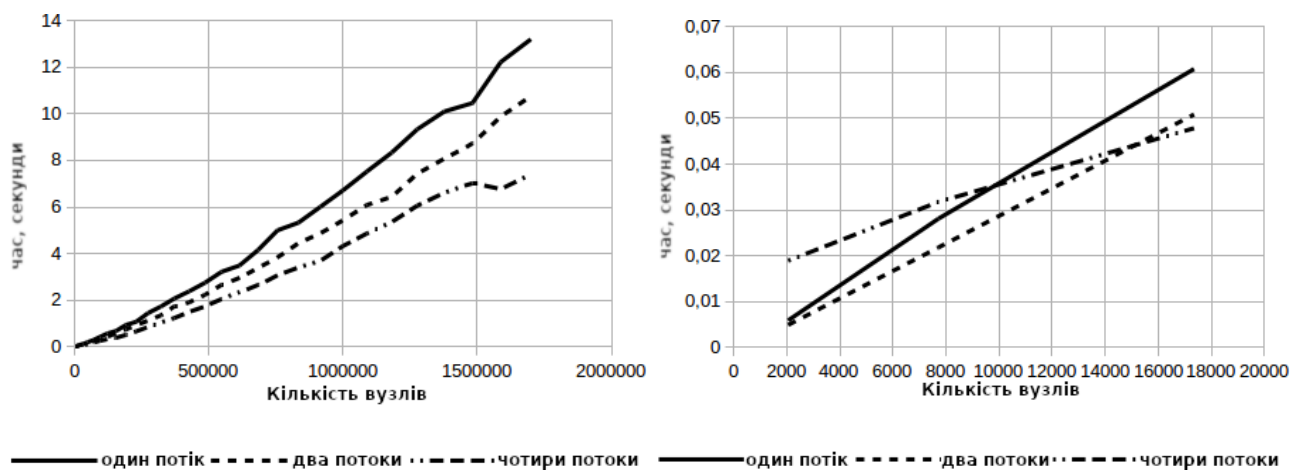
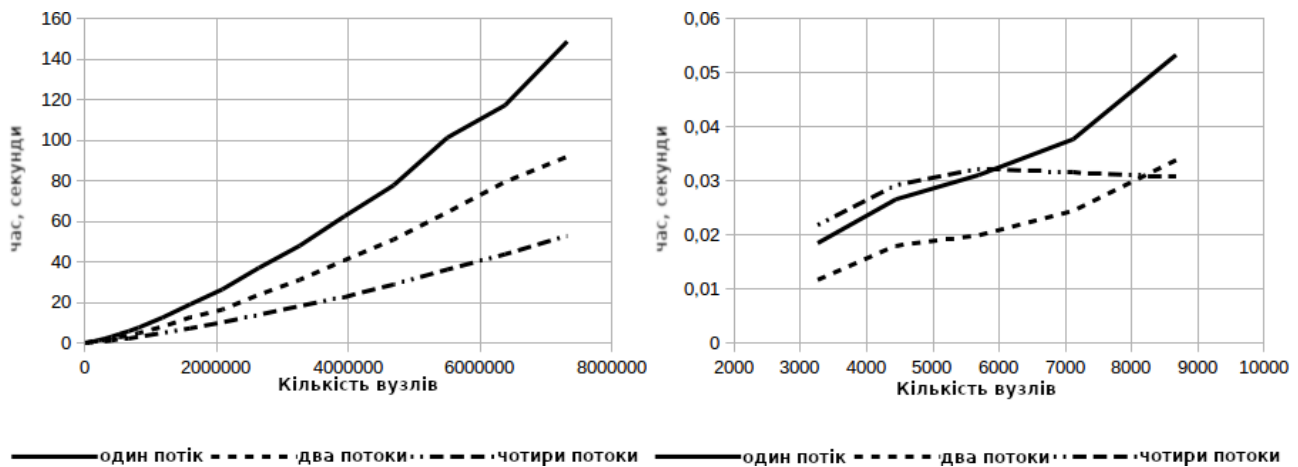


Рисунок 4.5 – Модель планки: залежність часу, витраченого на генерацію дискретної моделі від кількості вузлів у фоновій сітці

Чим складніша модель області, тим за меншої кількості вузлів паралельний алгоритм буде швидшим ніж послідовний. Наприклад, при побудові дискретної моделі прокладки (рис. 1.32) чотири паралельні потоки

будуть виконуватися швидше послідовного при 6000 вузлів і двох паралельних потоків при 8500 (рис. 4.6).



4.2.2 Скінченно-елементний аналіз з використанням технологій паралельних обчислень

Проектування сучасних деталей та конструкцій пов'язане з необхідністю використання чисельних методів для розв'язання задач високої розмірності. Врахування особливостей багатьох деталей і конструкцій обумовлює виникнення на підготовчих етапах складних моделей їх форми. Методи генерації дискретних моделей у результаті особливостей форми, умов задачі (наприклад, контакту, тріщин, багат шаровості тощо) та точності апроксимації може спричиняти велику кількість скінченних елементів. У підсумку для моделювання необхідні значні комп'ютерні ресурси та, як наслідок, часові витрати.

Природним рішенням задачі мінімізації потрібних комп'ютерних ресурсів є оптимізація кількості вузлів і елементів у дискретних моделях форм об'єктів. Проте, у випадках, коли зменшення кількості скінченних елементів зумовлює невиправдану втрату точності моделі, одним з шляхів оптимізації часових витрат є використання паралельних обчислень.

У методі скінченних елементів, якщо застосовують достатньо складні дискретні моделі форм об'єктів, етапи побудови глобальної матриці жорсткості та розв'язання системи лінійних алгебраїчних рівнянь є найбільш вимогливими щодо обчислювальних ресурсів. Отже, використання паралельних обчислень при виконанні цих етапів дозволить отримати найбільш вагомий результат з точки зору зменшення часу проектування та моделювання.

Першим кроком після етапів завантаження й ініціалізації вихідних даних паралельної реалізації метода скінченних елементів є визначення величини K – кількості наявних функціональних пристроїв (рис. 4.7). Ця величина є основою для обчислення кількості паралельних потоків. В мультипроцесорних системах при обчисленні глобальної матриці жорсткості операції вводу-виводу використовують тільки, якщо недостатньо основної пам'яті для зберігання вихідних даних або самої глобальної матриці. Тобто, за умови використання стисненого способу збереження глобальної матриці (яка, зазвичай, є розрідженою), час очікування операцій вводу-виводу буде мінімальним. Отже, на практиці величина K також визначає кількість паралельних комп'ютерних потоків.

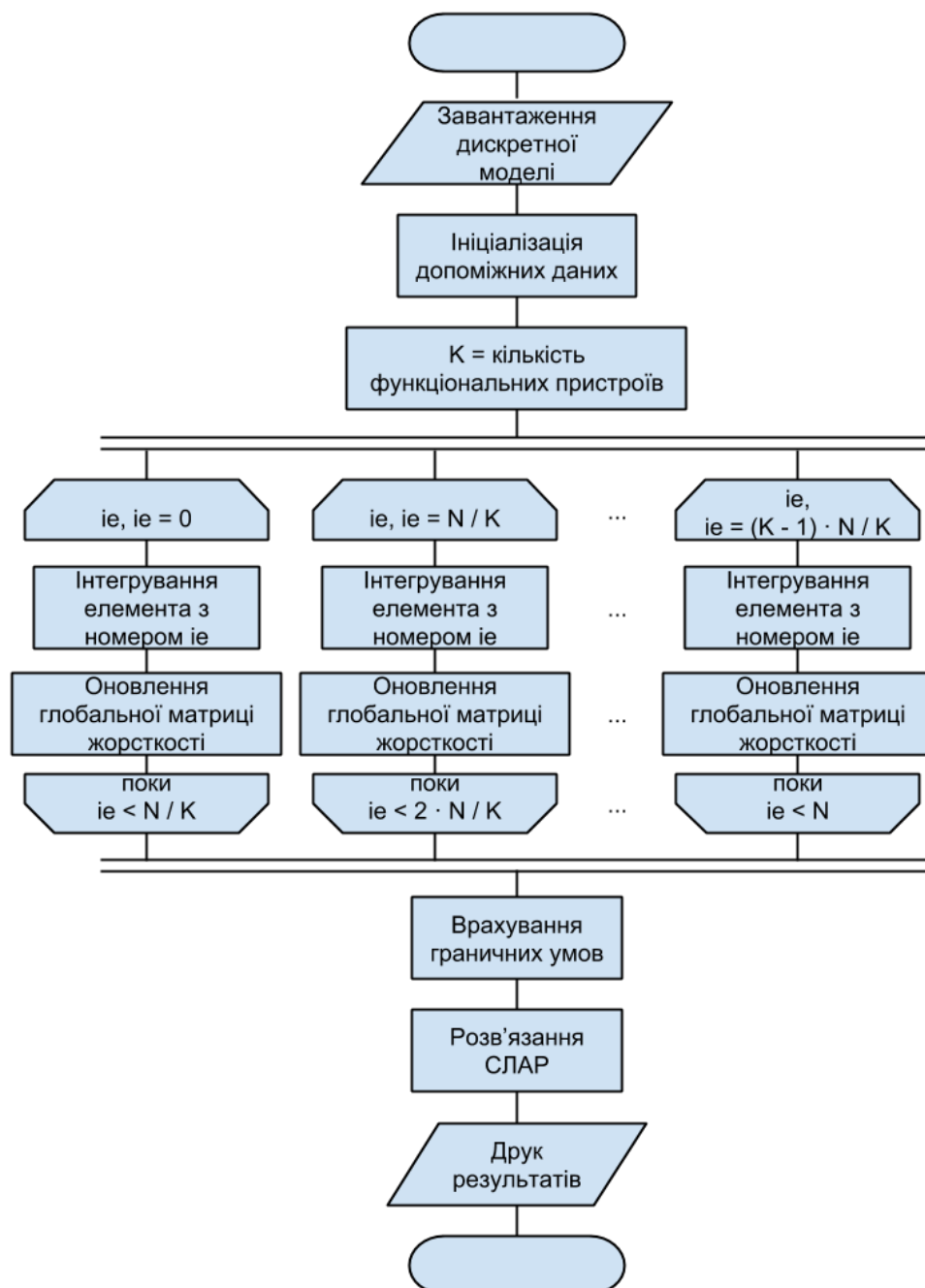


Рисунок 4.7 – Схема метода скінченних елементів на базі паралельних обчислень

Ансамблювання є додаванням за спеціальними правилами, отже, кожен з K паралельних потоків повинен виконувати ансамблювання N / K елементів (N – кількість елементів у дискретній моделі). Водночас всі дані (включно змінні циклів), які пов'язані з інтегруванням на скінченному елементі, повинні бути локальними для паралельних потоків (бути доступними тільки для потоку, що володіє; блок-схема зображена на рис. 4.7).

Крок оновлення глобальної матриці жорсткості, враховуючи стиснений формат збереження даних, є критичним з точки зору безпеки використання паралельних потоків. Отже, доступ іншим потокам до глобальної матриці жорсткості необхідно блокувати, поки один потік модифікує її дані.

Час, який необхідний для побудови глобальної матриці жорсткості за допомогою паралельного обчислювального процесу, можна оцінити величиною

$$T_p = N \left(\frac{I}{K} + m \right) + K t_{thread},$$

де I – процесорний час, який необхідний для інтегрування вихідного диференціального рівняння на одному скінченному елементі;

m – процесорний час, який необхідний для ансамблювання результатів інтегрування на одному елементі:

t_{thread} – процесорний час, який необхідний для ініціалізації одного комп'ютерного потоку.

Аналогічно, час, який необхідний для побудови глобальної матриці жорсткості на базі послідовного алгоритму, можна оцінити величиною

$$T_s = N(I + m).$$

Застосування паралельних обчислень має сенс, якщо $T_s > T_p$:

$$N(I + m) > N \left(\frac{I}{K} + m \right) + K t_{thread},$$

у результаті спрощення якої отримаємо таку оцінку:

$$N I > \frac{K^2}{K - 1} t_{thread}.$$

Отже, наприклад, для двох паралельних потоків, добуток кількості елементів на час інтегрування одного елемента має бути вчетверо більшим часу

ініціалізації паралельних потоків, що може не виконуватися при малій кількості скінченних елементів.

На практиці для розв'язання отриманих в результаті побудови глобальної матриці жорсткості СЛАР, зазвичай, використовують ітераційні методи (наприклад, метод спряжених градієнтів [338, 339] та інші). Їх ключовою особливістю є те, що результати поточної ітерації є вихідними даними для наступної. Отже, для скорочення часу пошуку розв'язку необхідно прискорювати внутрішні операції методу: добуток матриці на вектор, поелементний добуток векторів, обчислення нев'язки тощо.

З урахуванням того, що в результаті ансамблювання глобальна матриця жорсткості є розрідженою, схема обчислення добутку матриці на вектор в K паралельних потоків може отримати вигляд, наведений на рис. 4.8. Спільними для потоків даними є: матриця A , розмір якої $D \times D$, вектор x , вектор-результат r і число K . Оскільки обчислювальними потоками оновлюються різні елементи вектора r , то відсутня необхідність у блокуванні при синхронізації потоків. Лічильники циклів i та j , змінна t – локальні для кожного обчислювального потоку. У внутрішніх циклах по j обчислення виконуємо тільки для ненульових елементів i -го рядка матриці A .

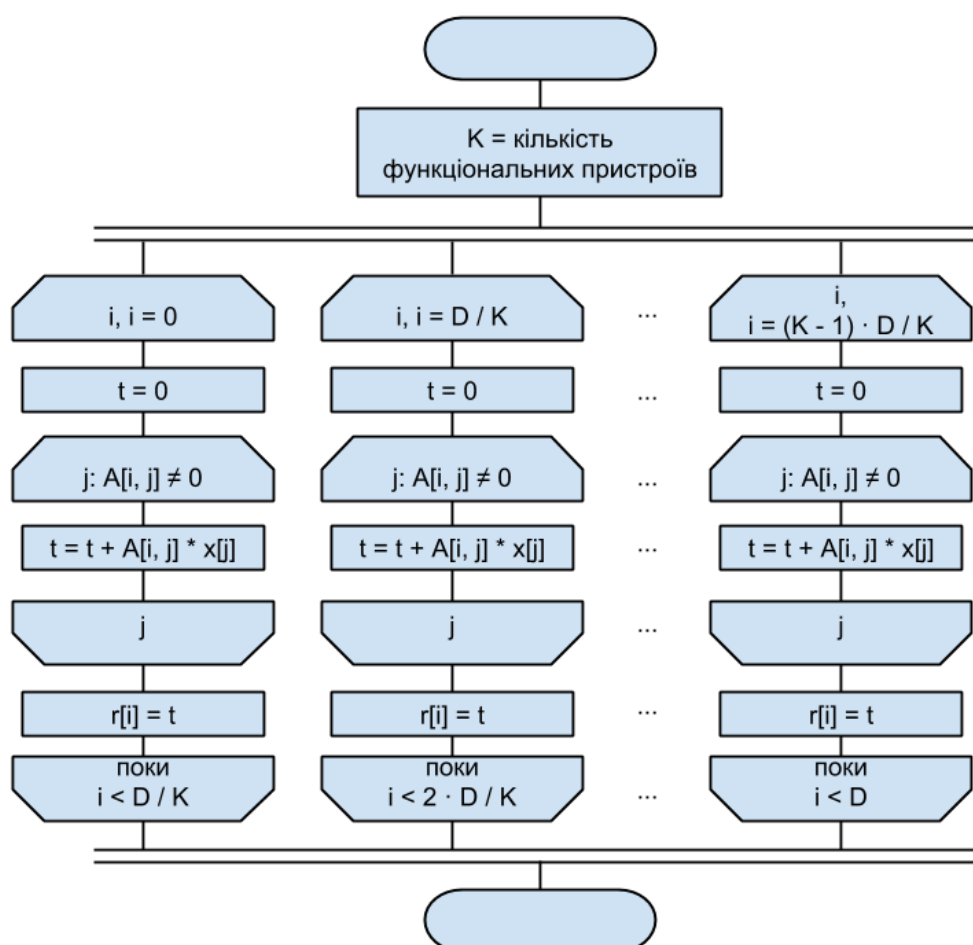


Рисунок 4.8 – Схема паралельного добутку розрідженої матриці та вектора

Як тестовий стенд використано програмно-апаратну конфігурацію з процесором Intel Core i3-380 (2.53 ГГц, 2 фізичних ядра + 2 віртуальних ядра) з 3 ГБ оперативної пам'яті під управлінням операційної системи openSUSE 12.2 (компілятор gcc 4.7), для маніпуляцій паралельними потоками – бібліотеки стандарту openMP.

Для оцінки часу кожен тестовий приклад послідовно запускаємо три рази та фіксуємо найменший витрачений час.

1. Задача Діріхле

Розглянемо розв'язок рівняння Лапласа з умовами Діріхле:

$$\frac{\partial U}{\partial x^2} + \frac{\partial U}{\partial y^2} + \frac{\partial U}{\partial z^2} = 0,$$

$$U_{\partial\Omega} = f(x, y, z).$$

Як область Ω задане зубчасте колесо (дискретна модель: 49500 вузлів, 40920 шестигранних скінченних елементів, рис. 4.9). Функція $f(x, y, z)$ має такий вигляд:

$$f(x, y, z) = \begin{cases} 0, & \text{на поверхні центрального отвору,} \\ 4 \sin^2(5\varphi), & \text{на поверхні зубців,} \end{cases}$$

де φ – кут повороту полярного радіуса до відповідної точки у площині Oxy .

Формула для обчислення локальної матриці $[K_e]$ отримає такий вигляд:

$$[K_e] = \iiint_{\Omega_e} \left(\begin{pmatrix} \frac{\partial N_1}{\partial x} \\ \vdots \\ \frac{\partial N_8}{\partial x} \end{pmatrix} \left\{ \frac{\partial N_1}{\partial x} \quad \dots \quad \frac{\partial N_8}{\partial x} \right\} + \begin{pmatrix} \frac{\partial N_1}{\partial y} \\ \vdots \\ \frac{\partial N_8}{\partial y} \end{pmatrix} \left\{ \frac{\partial N_1}{\partial y} \quad \dots \quad \frac{\partial N_8}{\partial y} \right\} + \right. \\ \left. + \begin{pmatrix} \frac{\partial N_1}{\partial z} \\ \vdots \\ \frac{\partial N_8}{\partial z} \end{pmatrix} \left\{ \frac{\partial N_1}{\partial z} \quad \dots \quad \frac{\partial N_8}{\partial z} \right\} \right) dx dy dz,$$

де N_i ($i = 1, \dots, 8$) – функції форми шестигранного скінченного елемента.

За результатами обчислювального експерименту отримано результати, що наведено в табл. 4.2 (час у секундах).

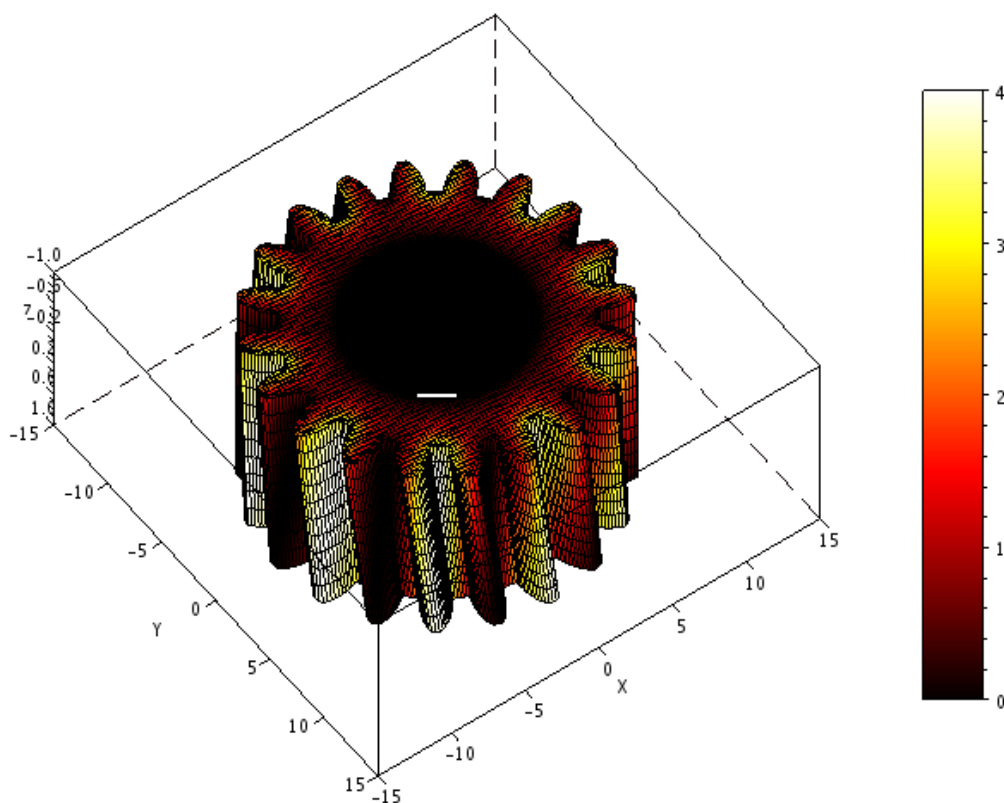


Рисунок 4.9 – Задача Діріхле: зубчасте колесо

Таблиця 4.2 – Розв'язання задачі Діріхле

Етап	Кількість потоків		
	1	2	4
	час, с		
Побудова глобальної матриці жорсткості	3,07639	1,56709	1,25949
Розв'язання СЛАР, 73 ітерації	0,304023	0,240893	0,264904

2. Дослідження прогину тришарової круглої пластинки, яка знаходиться під дією рівномірного навантаження (мікромодель)

Розглянемо затиснену по контуру тришарова кругла пластинка, радіус якої 0,4 м, товщина – 0,01 м. Внутрішній шар: товщина – 0,008 м, модуль Юнга – $E_i = 72017,3327$ МПа, коефіцієнт Пуассона – $\nu_i = 0,2999518536$. Зовнішні шари: товщина – 0,001 м, модуль Юнга – $E_o = 203200$ МПа, коефіцієнт Пуассона – $\nu_o = 0,27$. На на поверхню пластинки діє рівномірне поперечне навантаження $q = -0,05$ МПа.

Для дослідження побудована тривимірна дискретна модель сектора пластинки (рис. 4.10) на базі десяти шарів шестигранних скінченних елементів (59213 вузлів і 52240 елементів), товщина яких 0,001 м. Межі шарів матеріалу апроксимовані ребрами та гранями скінченних елементів.

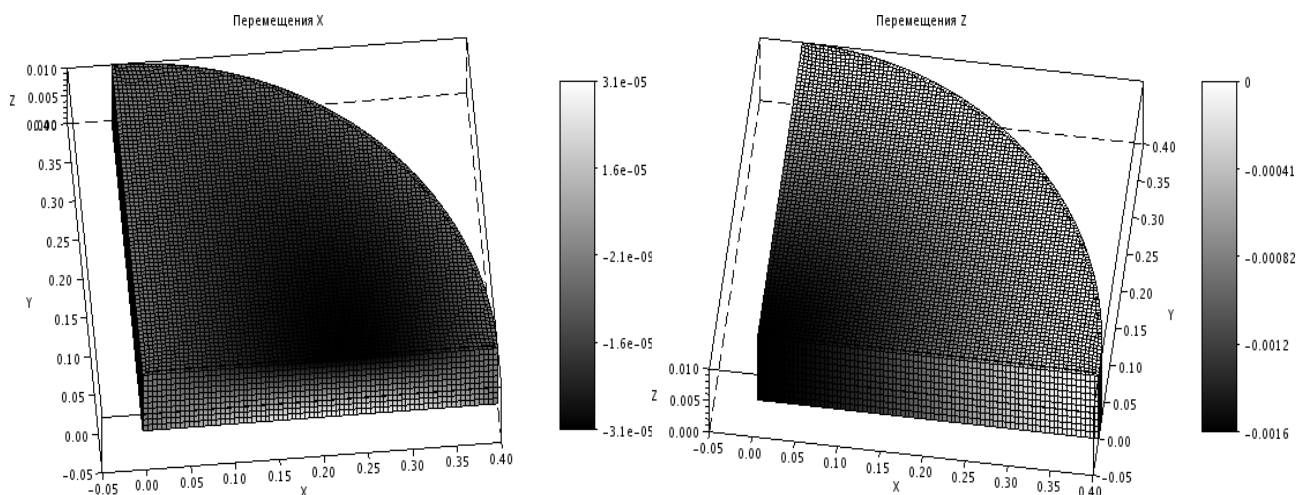


Рисунок 4.10 – Прогин тришарової пластинки

При обчисленні матриці $[D]$ у формулі (2.5) для врахування властивостей матеріалів шарів вважатимемо, що

$$E = \begin{cases} E_o, & z \geq 0,009 \vee z \leq 0,001, \\ E_i, & 0,001 \leq z \leq 0,009; \end{cases} \quad \nu = \begin{cases} \nu_o, & z \geq 0,009 \vee z \leq 0,001, \\ \nu_i, & 0,001 \leq z \leq 0,009. \end{cases}$$

Як результат обчислювального експерименту отримано результати, що наведені у табл. 4.3 (час у секундах).

Таблиця 4.3 – Дослідження пластинки

Етап	Кількість потоків		
	1	2	4
	час, с		
Побудова глобальної матриці жорсткості	83,7293	41,5063	31,7687
Розв'язання СЛАР, 6952 ітерації	298,287	191,299	189,134

Отже, обчислювальні експерименти показали, що використання паралельних обчислень в методі скінченних елементів дозволяє скоротити час на проектування та моделювання деталей та конструкцій. З таблиць 4.2 і 4.3 видно несуттєва перевага у часі при використанні чотирьох потоків відносно двох, що зумовлено двома віртуальними ядрами (які не мають відповідну фізичним ядрам швидкодію).

4.3. Проблемно-орієнтована мова моделювання форм об'єктів

Досить поширеним шляхом вирішення проблеми формалізації вихідних даних у системах інженерного аналізу є розробка лінгвістичного забезпечення – вхідних мов. Вхідна мова повинна включати в себе дві частини: мову опису об'єктів та мову опису завдань [340].

Мова опису об'єктів, в контексті цього дослідження, повинна надавати засоби опису моделей форм об'єктів, а також їх фізико-механічних характеристик і зовнішніх впливів (наприклад, прикладених сил). Мова опису завдань повинна включати засоби виконання певних проектних операцій та процедур: побудови дискретних моделей та чисельного аналізу параметрів стану об'єкта. Прикладами таких параметрів можуть бути умови зупинки алгоритмів.

4.3.1 Базовий синтаксис

Можливим розв'язком задачі створення лінгвістичного забезпечення для інженерного аналізу в САПР є розширення синтаксису, визначеного стандартом певної універсальної мови програмування, шляхом доповнення його специфічними для проблемної області мовними конструкціями. Водночас фактично буде створено нову проблемно-орієнтовану мову. Використання стандарту, на якому ґрунтуються популярні мови програмування, дозволяє спростити вивчення синтаксису проблемно-орієнтованої мови користувачами САПР. У даній роботі запропонована мова моделювання, що базується на стандарті ЕСМА-262 [341]. В основі стандарту ЕСМА покладено кілька вихідних технологій, найбільш відомою з яких є JavaScript. Серед його переваг можна виділити динамічну типізацію, яка дозволяє істотно скоротити код моделей.

Стандарт ЕСМА визначає специфікації об'єктно-орієнтованої мови для прикладних обчислень і маніпулювання обчислювальними об'єктами хост-оточення. Базовий синтаксис такої мови можна подати за допомогою розширеної форми Бекуса-Наура (рис.4.11). На рівні базового синтаксису програма є набором приписів і декларацій функцій, за допомогою яких можна визначати форми об'єктів і параметри розрахунку. Як алфавіт для ідентифікаторів можна використовувати латинські літери. Рядкові константи можна складати з довільного набору символів таблиці Unicode, які обмежені подвійними або одинарними лапками.

Згідно зі специфікацією, наведеною на рис. 4.11, кожний припис може бути блоком приписів, оголошенням змінних, виразом (простим, умовним,

перемиканням, поверненням або циклічним, зупинки ітерації або циклу), пустим виразам або пробілом. Якщо припис не є блоком, то він має закінчуватися символом вертикальної крапки з комою. Блоком приписів вважається деяка множина приписів, об'єднаних символами відкритої та закритої фігурних дужок.

```

програма = { припис | декларація функції } ;
припис = блок | пустий вираз | ( оголошення змінних, ';' ) | ( вираз, ';' ) | умовний
    вираз | циклічний вираз | стоп ітерація | стоп цикл | вираз повернення | вираз
    перемикання | пробіл ;
пустий вираз = ';' ;
блок = '{', { припис }, '}' ;
оголошення змінних = 'var', пробіл, ідентифікатор, [ '=', вираз ], { ',',
    ідентифікатор, [ '=', вираз ] } ;
вираз = виклик функції | декларація функції | ( ідентифікатор, присвоєння,
    ( обчислювальний вираз | виклик функції ) ) | створення об'єкта | інкремент |
    декремент ;
умовний вираз = 'if', '(', вираз, ')', припис, { 'else', припис } ;
циклічний вираз = ( 'do', припис, 'while', '(', вираз, ')', ';' ) |
    ( 'while', '(', вираз, ')', припис ) |
    ( 'for', '(', ( вираз, ';', вираз, ';', вираз ) | ( ініціалізація змінної, ';',
        вираз, ';', вираз ) | ( вираз, 'in', вираз ) | ( ініціалізація змінної, 'in'
        вираз ), ')', припис ) ;
стоп ітерація = 'continue', ';' ;
стоп цикл = 'break', ';' ;
вираз повернення = 'return', { вираз }, ';' ;
вираз перемикання = 'switch', '(', вираз, ')', '{'
    { 'case', вираз, ':', [ припис ] }, [ 'default', ':' [ припис ] ], '}' ;
обчислювальний вираз = простий вираз | умовний вираз | логічний вираз ;
умовний вираз = '(', вираз, ')', '?', вираз, ':', вираз ;
ініціалізація змінної = 'var', пробіл, ідентифікатор, '=', обчислювальний вираз ;
присвоєння = '=' | '+=' | '-=' | '*=' | '/=' ;
створення об'єкта = 'new', пробіл, ідентифікатор, ( вираз ) ;
декларація функції = 'function', { пробіл, ідентифікатор, } '(' [ ідентифікатор, [ { ',',
    ідентифікатор } ], ')', предписание ;
виклик функції = ідентифікатор, '(', [ вираз, [ { ',', вираз } ] ], ')' ;
інкремент = ('++', ідентифікатор) | (ідентифікатор, '++') ;
декремент = ('--', ідентифікатор) | (ідентифікатор, '--') ;
логічний вираз = лог_i [ { '||', лог_i } ] ;
лог_i = простий вираз, [ '&&', простий вираз ] ;
простий вираз = арифметичний вираз, [ оператор відношення, арифметичний вираз ] ;
оператор відношення = '<' | '<=' | '>' | '>=' | '==' | '!=' | '===' | '!==' ;
арифметичний вираз = [ '+', '-' ] терм { '+', '-' }, терм ;
терм = фактор { '*', '/', '%' }, фактор ;
фактор = ідентифікатор | константа | рядок | вираз | ('!', фактор) ;
константа = [ '+', '-' ] число [ '.' [ число ] [ ('e' | 'E') [ '+', '-' ] число ] ;
число = цифра { цифра } ;
цифра = '0' | '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9' ;
ідентифікатор = буква { буква | цифра | '_' } ;
рядок = ('"', { буква | цифра | символ }, '"') | (''', { буква | цифра | символ },
    ''') ;

```

Рисунок 4.11 – РБНФ базового синтаксису

Всі змінні та функції при такому підході розглядаються як об'єкти, для яких можна визначити поля і методи зі заданими областями видимості. Оголошення змінних починається з ключового слова `var`, за яким після пробільного символу вказуємо список змінних, розділених комами, з необов'язковою ініціалізацією через знак рівності. Якщо змінна є об'єктом, то вона обов'язково створюється з використанням оператора `new`. Для зберігання

чисел використовується 64-бітовий формат числа з рухомою комою IEEE 754 [342]. Рядкові значення зберігаються в форматі UTF-16.

Умовний вираз декларується з використанням оператора `if`, за яким в круглих дужках описується вираз, що визначає логічну гілку. Потім вказується припис, відповідний логічній гілці “істина”. Далі після оператора `else` можна зазначити припис, відповідний логічній гілці “хибність”.

Можна побачити, що можливе використання циклічних виразів з передумовою, з постумовою та циклів з лічильником (`for`). Також синтаксис дозволяє використовувати оператори логічного перемикачання (`switch`). Користувач може декларувати та викликати додаткові функції. Декларація функції здійснюється за допомогою ключового слова `function`, за яким наводиться необов'язковий ідентифікатор – ім'я функції (функції можуть бути анонімними), потім в круглих дужках вказується необов'язковий список параметрів, останнім вказується припис, відповідний тілу функції.

При оголошенні змінна розміститься в область видимості, яка відповідатиме операції в якій ця змінна оголошена. Якщо змінна оголошена поза функцією, то її область видимості вважається глобальною. Створення змінної в пам'яті відбувається при отриманні керування відповідною функцією (рис. 4.12).

```
var a; // a - undefined
var pi = 3.14; // pi ініціалізується значенням 3.14
function foo() {
    var a = 10; // a=10 - локальне в області видимості foo
}
function sqr(a) {
    return a * a;
}
var pi_2 = sqr(pi); // pi в квадраті
var arr = new Array(0, pi_2, 2); // створено масив з трьох елементів
print(arr[0]); // зневажувальний друк значення першого елемента (0)
```

Рисунок 4.12 – Змінні, функції та області видимості

Для створення масивів передбачений вбудований клас об'єктів `Array` (рис. 4.12, змінна `arr`), який дозволяє створювати масиви як заданої довжини, так і шляхом перерахування його елементів. Індексція елементів починається з нуля.

Отже, специфікації стандарту й описаний вище базовий синтаксис дозволяють визначити мову для маніпулювання об'єктами на базі примітивних типів. У підсистемі інженерного аналізу САПР машинобудування ці типи необхідно розширити високорівневими проблемно-орієнтованими абстракціями, які дозволять визначати такі об'єкти як точка, дискретна модель, параметри моделі, обчислювальний завдання тощо.

4.3.2 Допоміжні функції й операції

Допоміжні (арифметичні та тригонометричні) функції реалізовані в глобальному об'єкті Math, всі поля та методи якого статичні (для їх використання не потрібно створення екземпляра об'єкта). Методи даного об'єкта дозволяють обчислювати квадратний корінь (sqrt), натуральний логарифм (log), арккосинус (acos), арксинус (asin), арктангенс (atan, одного аргументу або відношення першого до другого аргументу, якщо їх два), синус (sin), косинус (cos), тангенс (tan), степінь (pow), модуль числа (abs), експоненту числа (exp), мінімум (min), максимум (max), найбільше ціле, що менше або дорівнює аргументу (floor), найменше ціле, що більше або дорівнює аргументу (ceil), округляти до найближчого цілого (round), генерувати випадкове число на відрізьку [0; 1] (random).

В об'єкті Math оголошені деякі відомі математичні константи: число Ейлера ($E \approx 2,71828$), натуральний логарифм 2 ($LN2 \approx 0,693$), логарифм за основою 2 числа Ейлера ($LOG2E \approx 1,442$), логарифм за основою 10 числа Ейлера ($LOG10E \approx 0,434$), число π ($PI \approx 3,14159$), значення $\frac{1}{\sqrt{2}}$ ($SQRT12 \approx 0,707$), $\sqrt{2}$ ($SQRT2 \approx 0,414$) і натуральний логарифм 10 ($LN10 \approx 2,302$).

При моделюванні часто виникає необхідність використання таких операцій, як сума або добуток значень деякої множини аргументів, приблизна перевірка на рівність тощо. Операції знаходження суми та добутку є багатомісними (рис. 4.13)

```
sum = 'sum', '(', вираз, [ '(', вираз ], ')';
prod = 'sum', '(', вираз, [ '(', вираз ], ')';
approx = 'approx', '(', вираз, '(', вираз, [ '(', вираз ], ')';
```

Рисунок 4.13 – РБНФ суми, добутку та прибливної рівності

Операція перевірки на приблизну рівність є тримісною (рис. 4.13): перші два аргументи – вирази для перевірки на приблизну рівність, третій необов'язковий аргумент – вираз, що задає точність перевірки. Математично ця операція подається такою формулою:

$$\text{approx}(x, y, \varepsilon) = \begin{cases} 1, & |x - y| < \varepsilon, \\ 0, & |x - y| \geq \varepsilon, \end{cases}$$

де ε – точність перевірки.

Отже, вираз $\text{аррох}(2.718281, 2.7183, 0.001)$ буде відповідати значенню 1 (“істина”), а вираз $\text{аррох}(2.718281, 2.7183, 0.00001)$ – значенню 0 (“хибність”).

4.3.3 Основні геометричні примітиви

Для логічного конструювання функціональних моделей форм об'єктів використовуються операції кон'юнкції, диз'юнкції та різниці. Відповідно, проблемно-орієнтована мова має дозволяти користувачеві визначати нові функціонально задані форми з використанням цих операцій.

Багатомісні операції кон'юнкції, диз'юнкції та різниці можна подати такими формулами:

$$\begin{cases} \text{con}(x_1, x_2, \dots, x_n) = x_1 \wedge x_2 \wedge \dots \wedge x_n, \\ \text{dis}(x_1, x_2, \dots, x_n) = x_1 \vee x_2 \vee \dots \vee x_n, \\ \text{diff}(x_1, x_2, \dots, x_n) = x_1 \wedge \neg x_2 \wedge \dots \wedge \neg x_n, \end{cases}$$

яким відповідають РБНФ, наведено на рис. 4.14.

```
con = 'con', '(', вираз, [ '(', вираз ], ')';
dis = 'dis', '(', вираз, [ '(', вираз ], ')';
diff = 'diff', '(', вираз, '(', вираз, [ '(', вираз ], ')';
```

Рисунок 4.14 – РБНФ логічних операцій з функціональними формами

Кожній формулі, що відповідає геометричному примітиву з розділу 2, можна протиставити однойменну функцію проблемно-орієнтованої мови з еквівалентною сигнатурою.

Наприклад, модель l-області, яка задана формулою (1.17), можна записати за допомогою проблемно-орієнтованої мови у такому вигляді:

```
var w = 30;
var h = 40;
var a = 10;
var count = 11;
function main(x, y) {
    return dis(rectangle(x - w / 2, y - a / 2, w, a), rectangle(x - a / 2, y - h / 2, a, h));
}
```

Тривимірну модель пробки (формула (1.33)) можна записати у такому вигляді:

```
function main(x, y, z){
    return con(cylinder(x, y, z, 25, 35), -con(-circle(x, y, 18), band(z - 17.5 + 25 - 2.5,
5)), -con(-circle(x, y, 21), band(z + 17.5 - 15 - 10, 20)), circle(x, y, 21 + 14.5 - z), -
con(regular(x, y, 22 / Math.sqrt(3), 6), band(z + 17.5 - 12.5, 25)));
}
```

Опис гібридних моделей здійснюється у два кроки. На першому кроці необхідно визначити відповідну до межі параметричну функцію. Водночас для визначення кривих і поверхонь Безьє передбачені відповідні примітиви.

Клас *Bezier* дозволяє визначити криву або поверхню Безьє на заданому наборі контрольних точок. Його конструктор має єдиний обов'язковий параметр – набір контрольних точок. Якщо параметром конструктора є масив двовимірних точок, то буде побудована крива. Для обчислення координати кривої, відповідної значенням параметра t передбачений метод класу *curve(t)*.

Якщо параметром конструктора класу *Bezier* буде матриця 4×4 , то результат – поверхня Безьє. Для обчислення точки на поверхні, що відповідає параметрам u і v передбачено метод *surface(u, v)*.

Для побудови дискретних моделей меж об'єктів передбачена функція-конструктор *BoundaryMesh*. Якщо у цій функції буде 4 параметри: кількість точок розбиття, початок і кінець інтервалу змін параметра, функція одного параметра, то буде побудована адаптивна дискретна модель для кривої. Якщо ж будуть передані 5 параметрів: число точок в напрямку u , число точок в напрямку v , початок і кінець інтервалу в першому напрямку, початок і кінець інтервалу в другому напрямку, функція двох параметрів, то буде побудована дискретна модель для параметричної поверхні.

Для перевірки належності точки простору до множини внутрішніх точок моделі форми об'єкту за допомогою тесту парності зі знаком в об'єктах класу *BoundaryMesh* передбачений метод *cfunction*. Параметрами цього методу є дво- або тривимірні координати.

Наприклад, такий код буде відповідати моделі профілю НАСА (рис. 1.46):

```
var bt = new Bezier(new Array(new Point(0, 0), new Point(1.14034, 2.44284), new
Point(11.1025, 3.65375), new Point(30.8809, 1.13309), new Point(39.9276,
0.09712), new Point(40, 0.0)));
var bb = new Bezier(new Array(new Point(0, 0), new Point(1.14034, -2.44284), new
Point(11.1025, -3.65375), new Point(30.8809, -1.13309), new Point(39.9276, -
0.09712), new Point(40, 0.0)));
function curve(t) {
    if (t <= 1.) return bt.curve(t);
    return bb.curve(2.0 - t);
};
```

```

var s = new BoundaryMesh(51, 0, 2.0, curve);
function main(x, y){
    return con(s.cfunction(x, y), -rectangle(x - 10, y, 6, 3), -circle(x - 20, y, 1.5), -
rectangle(x - 25, y, 2, 2));
}

```

4.3.4 Генерація дискретних моделей

Для забезпечення можливості генерації дискретних моделей в проблемно-орієнтованій мові передбачені відповідні вбудовані функції-конструктори. Найпростіші з них дозволяють генерувати рівномірні дискретні моделі.

Функція-конструктор *QuadrilateralMesh2D*(*m*, *n*, *Point*, *w*, *h*) повертає екземпляр дискретної моделі на базі чотирикутників, яка складається з $m \times n$ вузлів, для прямокутника, нижній лівий кут якого заданий координатами *Point*, ширина – *w*, висота – *h*. Функція-конструктор *TriangularMesh2D* з такими ж параметрами поверне еквівалентний екземпляр на основі трикутників.

У тривимірному випадку конструктор *HexahedralMesh*(*m*, *n*, *k*, *Point*, *w*, *h*, *d*) повертає екземпляр дискретної моделі на базі шестигранників, яка складається з $m \times n \times k$ вузлів, для прямокутної призми дальній нижній лівий кут, якої заданий координатами *Point*, ширина – *w*, висота – *h*, глибина – *d*. Конструктор *TetrahedralMesh* з такими ж параметрами поверне еквівалентний екземпляр на основі тетраедрів.

Зазначимо, що конструктори *QuadrilateralMesh2D*, *TriangularMesh2D*, *HexahedralMesh* і *TetrahedralMesh*, дозволяють конвертувати дискретні моделі форм об'єктів. Для цього єдиним параметром цих функцій повинен бути екземпляр дискретної моделі для конвертації.

Для генерації дискретних моделей меж об'єктів передбачено функцію-конструктор *BoundaryMesh*. Якщо функції *BoundaryMesh* передати два параметри: екземпляр дискретної моделі та неявну функцію області, то за допомогою алгоритму *background_grid_surface* буде згенерована дискретна модель межі, яка задана неявною функцією. Якщо у функції *BoundaryMesh* тільки один параметр – дискретна модель, то її результатом буде множина межових граней. Наприклад, для кубика, який заданий формулою (1.58), дискретну модель його межі (тривимірні трикутники) на базі фонові сітки тетраедрів (з конвертацією з шестигранників, рис. 1.53) можна описати таким кодом:

```
SLEVEL = 2;
```

```

OLEVEL = 8;
function quad(x, y, z){
    return con(cuboid(x, y, z, 2, 2, 2), -circle(x, y, 0.5), -circle(y, z, 0.5));
}
var n = 21;
var background = new TetrahedralMesh(new HexahedralMesh(n, n, n, new Point(-1,
-1, -1), 2, 2, 2));
var mesh = new BoundaryMesh(background, quad);

```

У наведеному вище прикладі глобальні параметри SLEVEL і OLEVEL використовуються, відповідно, для обмеження кількості ітерацій згладжування Лапласа і покращення дискретних моделей на основі мінімізації функціоналу відстані-довжини (піункт 1.10.2.1).

Для генерації дискретної моделі з використанням узагальненого методу для елементів типу солід (алгоритм *background-grid-mesh*, пункт 1.11.2) необхідно передбачено функцію *BackgroundGridMesh* з двома параметрами: екземпляром фонові дискретної моделі та неявною функцією, що подає форму об'єкта. Наприклад, для кубика, форма якого подана формулою (1.58), дискретну модель на базі шестигранників можна описати таким кодом:

```

SLEVEL = 4;
OLEVEL = 8;
function quad(x, y, z){
    return con(cuboid(x, y, z, 2, 2, 2), -circle(x, y, 0.5), -circle(y, z, 0.5));
}
var n = 21;
var b = new HexahedralMesh(n, n, n, new Point(-1, -1, -1), 2, 2, 2);
var mesh = new BackgroundGridMesh(b, quad);

```

При генерації дискретних форм об'єктів на базі солідів глобальні параметри SLEVEL і OLEVEL використовуються для, відповідно, обмеження кількості ітерацій згладжування Лапласа внутрішніх вузлів та оптимізації дискретних моделей на основі локальної мінімізації функціоналу площі (або обсягу) зі знаком.

Для спрощення генерації дискретних моделей найбільш поширених типів оболонок у проблемно-орієнтованій мові передбачені такі функції: *CylinderTriangles(n, m, r, h)* і *CylinderQuads(n, m, r, h)*, *ConeTriangles(n, m, r1, r2, h)* і *ConeQuads(n, m, r1, r2, h)*. Перша пара призначена для створення рівномірних дискретних моделей (на базі трикутників або чотирикутників) для циліндричних поверхонь ($n \times m$ вузлів, радіуса – r і висота – h). Друга пара – для рівномірних дискретних моделей конічних поверхонь з

$n \times m$ вузлів, з радіусами основ $r1$ і $r2$, висотою h . Якщо до цих параметрів додати функцію, яка описує форму отворів, то результатом буде дискретна модель, побудована у полі значень параметрів відповідної поверхні, з адаптацією до меж, заданих цією функцією. Наприклад, для конічної оболонки зі складною конфігурацією отворів (рис. 1.56) буде така формальна модель:

```

var r1 = 3.00 / 2.0;
var r2 = 3.98 / 2.0;
var h = 4.014;
var m = 80;
var n = ceil(m * 2 * PI * r1 / h);
var h1 = h - 0.580;
var h2 = h - 0.485;
var h3 = h - 0.460;
var h4 = 0.52;
var h5 = 0.72;
function holes(x, y, z){
    var xx1 = x * cos(-0.65) + z * sin(-0.65);
    var xx2 = x * cos(-1.31) + z * sin(-1.31);
    var xx3 = x * cos(-2.09) + z * sin(-2.09);
    var xx4 = x * cos(-1.31) + z * sin(-1.31);
    var zz4 = -x * sin(-1.31) + z * cos(-1.31);
    var xx5 = x * cos(2.09) + z * sin(2.09);
    var zz5 = -x * sin(2.09) + z * cos(2.09);
    var xx6 = x * cos(0.65) + z * sin(0.65);
    var zz6 = -x * sin(0.65) + z * cos(0.65);
    var xx7 = x * cos(-2.43) + z * sin(-2.43);
    var zz7 = -x * sin(-2.43) + z * cos(-2.43);
    return -dis(rectangle(xx1, y - h1, 0.40, 0.40, 0.055), rectangle(xx2, y - h1, 0.40, 0.40, 0.055), rectangle(xx3, y - h2, 0.40, 0.25, 0.065), con(rectangle(xx4, y - h4, 0.68, 0.45, 0.075), zz4), con(rectangle(xx5, y - h4, 0.68, 0.45, 0.075), zz5), con(rectangle(xx6, y - h5, 0.44, 0.88, 0.075), zz6), con(circle(xx7, y - h3, 0.105), zz7));
}
var mesh = ConeTriangles(n, m, r1, r2, h, holes);

```

Для побудови дискретних моделей передбачена можливість об'єднання частин. У кожного екземпляру дискретної моделі є метод *add*, єдиним параметром якого є екземпляр іншої дискретної моделі. Водночас метод *translate* дозволяє перемістити всі вузли моделі на заданий вектор. У результаті об'єднання моделей вузли з однаковими координатами матимуть однакові коди. Наприклад, оболонку як комбінацію конуса та циліндра (рис. 4.15) можна подати у вигляді такого кода:

```

var pi = Math.PI;
var count = 50;
var cl = 5.912 - 0.070; // довжина циліндричної частини
var l = 3.1+6.012-5.912; // довжини конічної частини
var r1 = 0.536/2; // радіус меншої основи конічної частини
var r5 = 2.4/2; // радіус більшої основи конічної частини, радіус циліндричної частини
var mesh = new ConeQuads(count, count, r1, r5, l);
var c1 = new CylinderQuads(count, Math.ceil(count * cl / (2 * pi * r5)), r5, cl);
c1.translate(0, l, 0); // переміщення циліндра на радіус-вектор з координатами (0; l; 0)
mesh.add(c1);

```

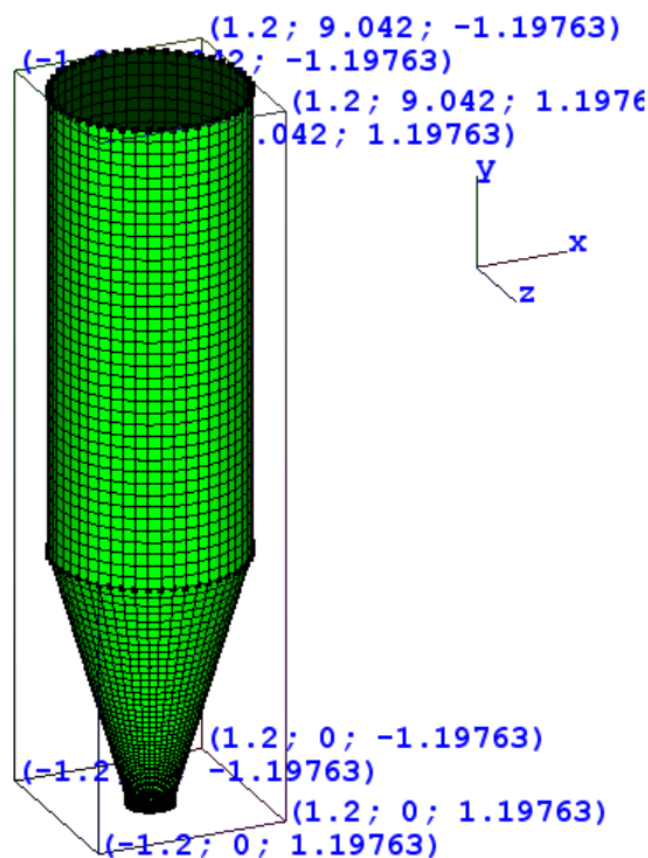


Рисунок 4.15 – Дискретна модель оболонки “Конус-Циліндр”

4.3.5 Опис скінченно-елементних задач

Після генерації дискретної моделі форми об’єкта для скінченно-елементного аналізу необхідно визначити крайові умови та вплив зовнішнього середовища. Як показано у підрозділі 7.1 крайові умови та впливи зовнішнього середовища можуть бути екземплярами одного базового класу *FeaValue* (рис. 4.4).

Для задання крайових умов передбачена функція-конструктор *BoundaryCondition(direction, select, value)* з трьома параметрами. Перший параметр – напрям дії *direction*, який може бути одним з таких значень: ALL (всі), FIRST (перше), SECOND (друге), THIRD (третє), FOURTH (четверте), FIFTH (п'яте), SIXTH (шосте), SEVENTH (сьоме), EIGHTH (восьме), NINTH (дев'яте), TENTH (десяте). Другий параметр – функція координат *select*, що повертає істину для точок, в яких діє певна умова (цей параметр необов'язковий, якщо він не заданий, то вважається, що умова діє на всі точки). Останній параметр – функція координат або значення *value* (якщо воно константне), які відповідні крайовій умові.

Для задання навантажень передбачені конструктори *NodalForce* (вузлові навантаження), *SurfaceForce* (поверхневі навантаження) і *VolumeForce* (об'ємні сили). Їх параметри синтаксично та семантично еквівалентні параметрам функції *BoundaryCondition* (результати є об'єктами спільного базового класу).

Для двовимірного скінченно-елементного аналізу передбачена функція *Fem2D(mesh, h, D, ...)*, обов'язковими трьома параметрами якої (за сигнатурою) є екземпляр дискретної моделі на базі двовимірних елементів *mesh*, товщина об'єкта *h*, матриця зв'язків між напруженнями та деформаціями *D*. Далі можна перерахувати довільна кількість крайових умов і навантажень.

Наприклад, задачу аналізу напружено-деформованого стану фланця з підрозділу 3.2 (рис. 3.8) можна описати у такому вигляді:

```
function main(x, y) {
    return con(dis(rectangle(x, y, 20, 0, 80, 100), rectangle(x, y, 0, 20, 100, 60),
    circle(x - 20, y - 20, 20), circle(x - 20, y - 80, 20), circle(x - 100, y - 50, 50)),
    -circle(x - 100, y - 50, 30), -circle(x - 20, y - 20, 10), -circle(x - 20, y - 80, 10));
    }
    SLEVEL = 4;
    OLEVEL = 8;
    var c = new Array(new Point(100, 20));
    var b = new QuadrilateralMesh2D(1.5 * 31, 31, new Point(0, 0), 150, 100);
    var mesh = new BackgroundGridMesh(b, main);
    var E = 200000;
    var nu = 0.33;
    var q = 20 * 20;
    var fixed = new BoundaryCondition( ALL,
        function (x, y){
            if (approx(circle(x - 20, y - 20, 10), 0.0) || approx(circle(x - 20, y - 80, 10), 0.0))
                return true;
            return false;
        },
```

```

0);
var force = new SurfaceForce( FIRST,
    function (x, y){
        if (approx(circle(x - 100, y - 50, 30), 0.0)) return true;
        return false;
    },
    q);
Fem2D(mesh, 20, new PlaneStressMatrix(E, nu), fixed, force);

```

У наведеному вище прикладі функція-конструктор *PlaneStressMatrix*(*E*, *nu*) створює матрицю зв'язків між напруженнями та деформаціями (плоско-напружений стан) для заданих модуля Юнга *E* та коефіцієнта Пуассона *nu*. Аналогічно, *PlaneStrainMatrix* створює матрицю зв'язків для плоско-деформованого стану.

Для аналізу напружено-деформованого стану у тривимірному випадку передбачена функція *Fem3D*(*mesh*, *D*, ...) з двома обов'язковими параметрами. Перший параметр – екземпляр тривимірної дискретної моделі форми об'єкта на базі тривимірних солідів. Другий параметр – матриця зв'язків між напруженнями та деформаціями у тривимірному випадку. Починаючи з третього параметра можна перераховувати крайові умови і навантаження.

Для створення матриці зв'язків між напруженнями та деформаціями у тривимірному випадку передбачено функцію-конструктор *VolumeStressMatrix*(*E*, *nu*), де *E* – модуль Юнга, *nu* – коефіцієнт Пуассона. Наприклад, аналіз напружено-деформованого стану стрижня під дією власної ваги з параметрами як у пункті 2.1.2 можна описати з використанням такого коду:

```

var l = 20;
var c = 1;
var E = 210000;
var nu = 0.3;
var gamma = 0.076500; // питома вага
var nx = 3;
var nz = (nx - 1) * 20 + 1;
var mesh = new HexahedralMesh(nx, nx, nz, new Point(-c/2, -c/2, 0), c, c, l);
var fixed = new BoundaryCondition( THIRD,
    function (x, y, z){
        if (approx(z, l)) return true;
        return false;
    }, 0);
var force = new VolumeForce( THIRD, -gamma);
Fem3D(mesh, new VolumeStressMatrix(E, nu), fixed, force);

```

Додатково передбачено функції *MindlinPlate* і *MindlinShell*, які можна використати для дослідження, відповідно, пластин (на базі двовимірних елементів) і оболонок (на базі тривимірних трикутників або чотирикутників) на базі теорії деформацій зсуву другого порядку (узагальнення теорії Тимошенко-Міндліна), параметри цих функцій ідентичні тим, що використано у *Fem2D*. Але товщина у цих двох функціях може бути не константною, а функцією координат. Також, якщо товщина та матриця зв'язків між напруженнями та деформаціями є масивами знаень, то використовується відповідна теорія для багат шарових пластин і оболонок.

Для всіх функцій скінченно-елементного аналізу можна активувати адаптивність. Для управління адаптивним скінченно-елементним аналізом передбачено три глобальні параметри: ADAPT, ALEVEL і ADELTA. Якщо параметр ADAPT дорівнює true, то використовується адаптивний аналіз. Водночас параметр ALEVEL є цілим числом – обмеженням на максимальну кількість ітерацій, а параметр ADELTA – на точність. За замовченням ADAPT=false, ALEVEL=1, ADELTA=1. Наприклад, для адаптивного аналізу напружено-деформованого стану оболонки-комбінації циліндра та конуса (вихідна дискретна модель зображена на рис. 4.15) можна використати такий код:

```
var cylinder_fix = new BoundaryCondition(ALL,
    function (x, y, z) {
        if (approx(y, cl + l)) return true;
        return false;
    }, 0);
var axial = new SurfaceForce(SECOND,
    function (x, y, z) {
        if (approx(y, 0)) return 0.48 * 9806.65 * 1.0E-6;
        return 0;
    });
var q = 0.5 * 98066.5 * 10.0E-6;
var sl = Math.sqrt(l*l + r5*r5);
var force_x = new ShellDistributedForce(FIRST,
    function (x, y, z) {
        if (y <= l) return -q * (l / sl) * Math.cos(Math.atan2(z, x));
        return -q * Math.cos(Math.atan2(z, x));
    });
var force_y = new ShellDistributedForce(SECOND,
    function (x, y, z) {
        if (y <= l) return q * (r5 - r1) / sl;
        return 0;
    });
```

```

var force_z = new ShellDistributedForce(THIRD,
    function (x, y, z) {
        if (y <= l) return -q * (l / sl) * Math.sin(Math.atan2(z, x));
        return -q * Math.sin(Math.atan2(z, x));
    });
function thickness(x, y, z){
    if ((l - 0.025 <= y && y <= l + 0.025) || l + cl - 0.025 <= y)
        return 0.0418;
    if ((l - 0.1 <= y && y <= l + 0.1) || l + cl - 0.1 <= y)
        return 0.0358;
    return 0.0282;
}
ADAPT = true; ALEVEL = 3; ADELTA = 0.2;
MindlinShell(
    mesh, thickness,
    new PlaneStressMatrix(34000, 0.3),
    cylinder_fix, axial, force_x, force_y, force_z);

```

У наведеному вищі прикладі крайова умова *cylinder_fix* буде застосована до всіх вузлів, у яких координата *y* дорівнює *cl+l* (відповідним вузлам заборонено переміщення у всіх напрямках). Сила *axial* діє тільки на сторони елементів, для яких *y=0* (менший радіус конічної частини), та відповідає осьовому стисненню. Сили *force_x*, *force_y* і *force_z* моделюють дію рівномірного зовнішнього тиску (вони застосовують до всіх елементів). Товщина є функцією координат оболонки, вона збільшується на з'єднанні конічної та циліндричної частин і біля закріпленої основи циліндричної частини (результати інтерпретації цього коду наведені на рис. 4.16).

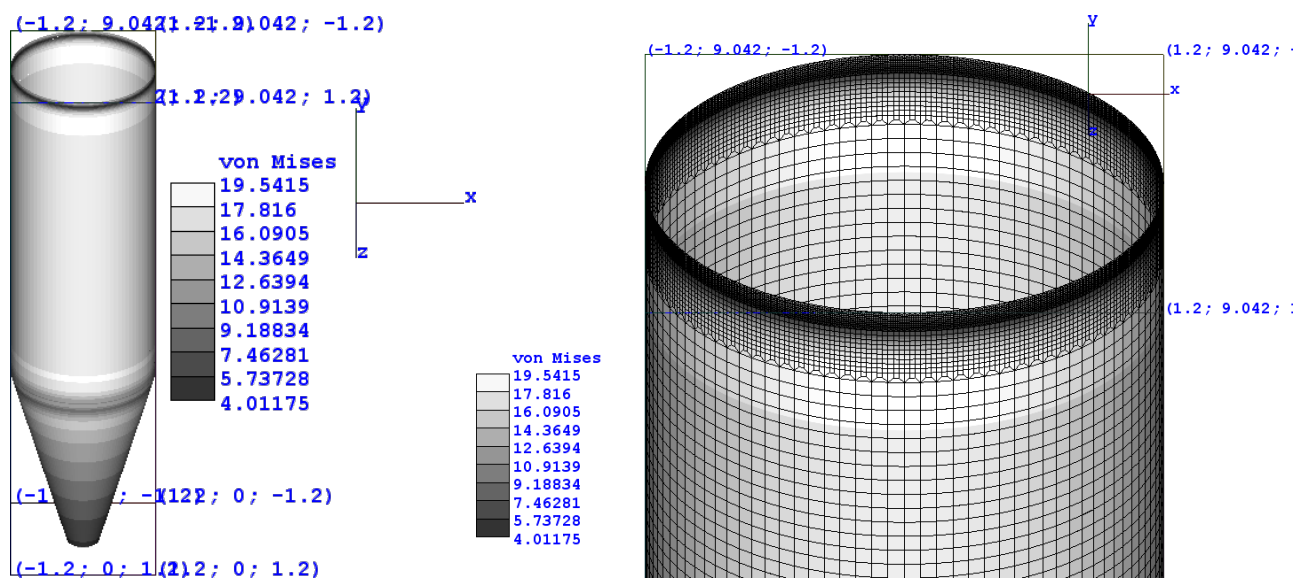


Рисунок 4.16 – Напружено-деформований стан оболонки «Конус-Циліндр»

4.4. Об'єктно-орієнтована модель САПР для паралельних комп'ютерних систем

При розробці програм, що підтримують паралельні розрахунки, може бути ефективним застосування шаблону проектування Prototype (Прототип) [303], який дозволяє копіювати об'єкти будь-якої складності, не вдаючись у подробиці їх програмної реалізації. Так, при застосуванні цього шаблону для розробки програми скінченно-елементного аналізу можна, наприклад, створювати точні копії об'єктів, що описують побудову локальної матриці жорсткості скінченного елемента на певному вузлі обчислювального кластера або процесорі, без прив'язки до їх класу (рис. 4.17). Паралельну САПР машинобудування можна подати як агрегацію абстрактних процесорів МСЕ (FeaProblem), конкретні реалізації яких дозволяють розв'язувати певні типи задач (наприклад, аналізу напружено-деформованого стану суцільних тіл, оболонок тощо). Після створення копії FeaProblem у хмарній САПР відповідний об'єкт реєструється як готовий до використання у паралельній системі (призначається локальний або віддалений процесор для виконання команд копії об'єкта).

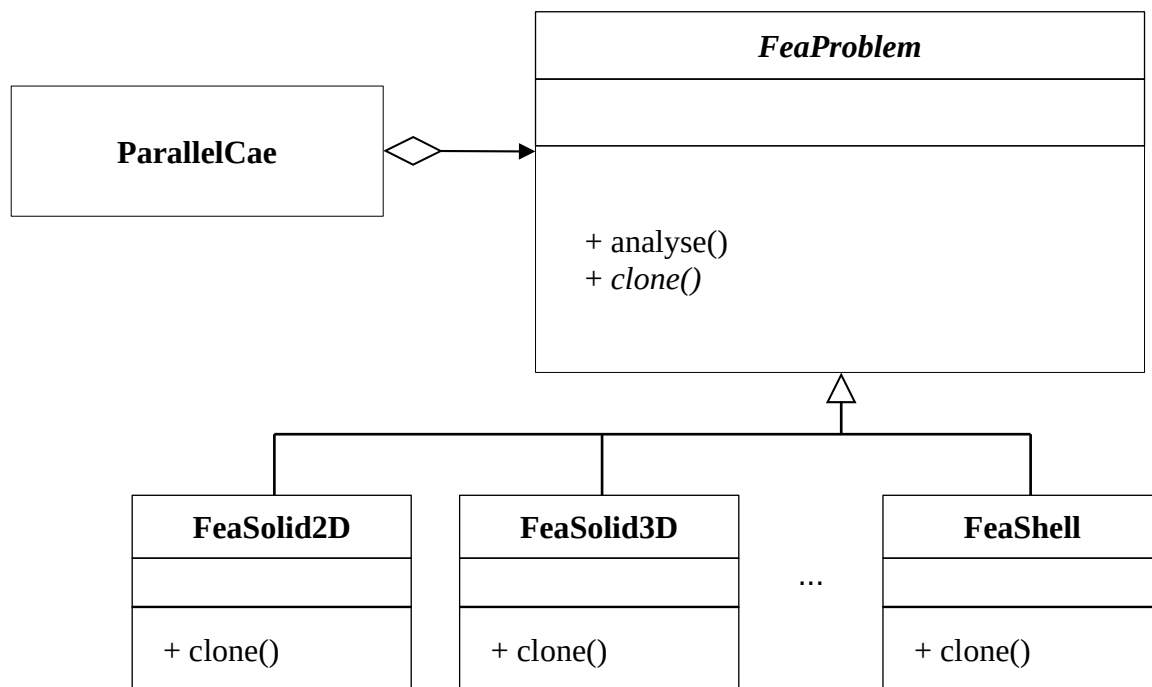


Рисунок 4.17 – Модель хмарної САПР на базі шаблону Prototype

Головною перевагою застосування цього шаблону проектування при розробці розподіленого програмного забезпечення є легкість створення нових паралельно працюючих розрахункових об'єктів (потоків виконання) на

обчислювальних вузлах та можливість гнучкого керування ними. Саме тому застосування шаблону Prototype при розробці процесора скінченно елементного аналізу, який підтримує паралельні розрахунки, є перспективним.

Аналогічним чином можна спроектувати класи, що реалізують паралельне врахування крайових умов; пряме або ітераційне розв'язання СЛАР тощо. Твірною фабрикою (Prototype Factory) у структурі, наведеній на рис. 4.17 може бути ParallelCae (або його нащадки), який у процесі роботи алгоритму МСЕ клонує розрахункові об'єкти на наявних обчислювальних міцностях (ядрах, процесорах, обчислювальних вузлах тощо) для розпаралелювання процесу розрахунку.

4.5. Модель роботи методів у хмарному середовищі

Менеджер черг є проміжним модулем програми, який передає повідомлення з протоколу обміну повідомленнями відправника до протоколу обміну повідомленнями приймача. Менеджери черг – це елементи в телекомунікаційних або комп'ютерних мережах, де зв'язуються програмні додатки, обмінюючись визначеними повідомленнями. Менеджери черг є будівельним блоком інформаційного проміжного програмного забезпечення.

Менеджер черг є архітектурним шаблоном для підтвердження повідомлень, перетворення та маршрутизації [343]. Він опосередковує комунікацію між додатками, мінімізуючи взаємне усвідомлення того, що програми повинні мати один одного, щоб мати можливість обмінюватися повідомленнями, ефективно відокремлюючи різні компоненти системи.

З підвищенням вимог до хмарних додатків ускладнюється архітектура хмарних систем. Доводиться вдаватися до створення хмар: кілька серверів об'єднують у кластер; навантаження між ними розподіляється за допомогою комплексу спеціальних методів балансування навантаження. Крім вирішення проблеми високих навантажень комп'ютерна хмара допомагає також забезпечити резервування серверів один на одного.

Ефективність хмарних обчислень безпосередньо залежить від того, як розподіляється (балансується) навантаження між елементами кластера [344].

Розподіл навантаження на таких складних системах є викликом для архітекторів програмного забезпечення. У таких випадках проявляється ефективність та простота використання менеджерів черг в якості каналу зв'язку між сервісами системи або вузлами кластера.

4.5.1 Лінійна модель менеджера черг

Черги повідомлень, по суті, є сполучною ланкою між різними процесами у додатках і забезпечують надійний і масштабований інтерфейс взаємодії з іншими підключеними системами і пристроями.

Черга – це структура даних з дисципліною доступу до елементів «перший прийшов – перший вийшов». Додавання елемента можливо лише в кінець черги, вибірка – тільки з початку черги, при цьому обраний елемент з черги видаляється [345].

Перший спосіб управління чергами у хмарі можна подати у вигляді масиву і двох цілочисельних змінних `start` і `end` (рис. 4.18).

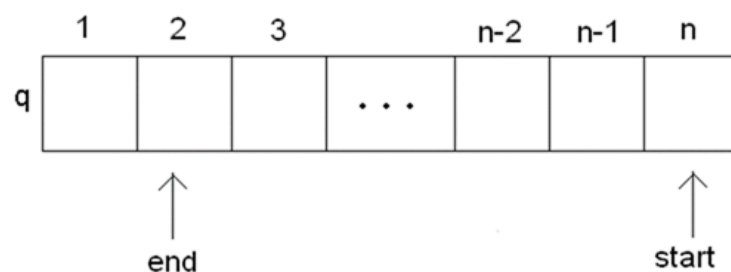


Рисунок 4.18 – Модель черги за допомогою масиву

Зазвичай `start` вказує на голову черги, `end` – на елемент, який заповниться, коли в чергу увійде новий елемент. При додаванні елемента в чергу в `q[end]` записується новий елемент черги, а `end` зменшується на одиницю. Якщо значення `end` стає менше 1, то ми як би циклічно обходимо масив, і значення змінної стає рівним `n`. Витяг елемента з черги проводиться аналогічно: після вилучення елемента `q[start]` з черги змінна `start` зменшується на 1. З такими алгоритмами один осередок з `n` завжди буде незайнятим (так як чергу з `n` елементами неможливо відрізнити від порожньої), що компенсується простотою алгоритмів.

Переваги даного методу: можлива незначна економія пам'яті в порівнянні з другими способами; простіше в розробці.

Недоліки: максимальна кількість елементів в черзі обмежена розміром масиву. При його переповненні потрібно перевиділення пам'яті і копіювання всіх елементів в новий масив.

Другий спосіб заснований на роботі з динамічною пам'яттю. Чергу представляється в якості лінійного списку, в якому додавання/видалення елементів йде строго з відповідних його кінців.

Переваги даного методу: розмір черги обмежений лише обсягом пам'яті. Недоліки: складніша розробка, необхідність більшого обсягу пам'яті, при роботі з такою чергою, пам'ять сильніше фрагментується, більш повільна робота з чергою.

4.5.2 Модель менеджера черг у розподіленій комп'ютерній системі

Одним з найбільш поширених способів реалізації хмарних середовищ базується на використанні обчислювальних кластерів. Під обчислювальним кластером, зазвичай, розуміють групу комп'ютерів, об'єднаних спеціальним мережевим обладнанням, які здатні обмінюватись повідомленнями протягом вирішення спільного завдання або виконання запитів від клієнтів.

Для хмарної реалізації САПР необхідно забезпечити можливість розподілу завдань між вузлами кластера. Типовий алгоритм роботи хмарної САПР може бути таким: 1) клієнт ініціалізує завдання через web-, мобільний або desktop-додаток; 2) менеджер черг або брокер обирає вільний або найменш завантажений вузол кластера, який починає виконання завдання; 3) вузол після закінчення роботи надсилає результати клієнту через брокера (рис. 4.19).

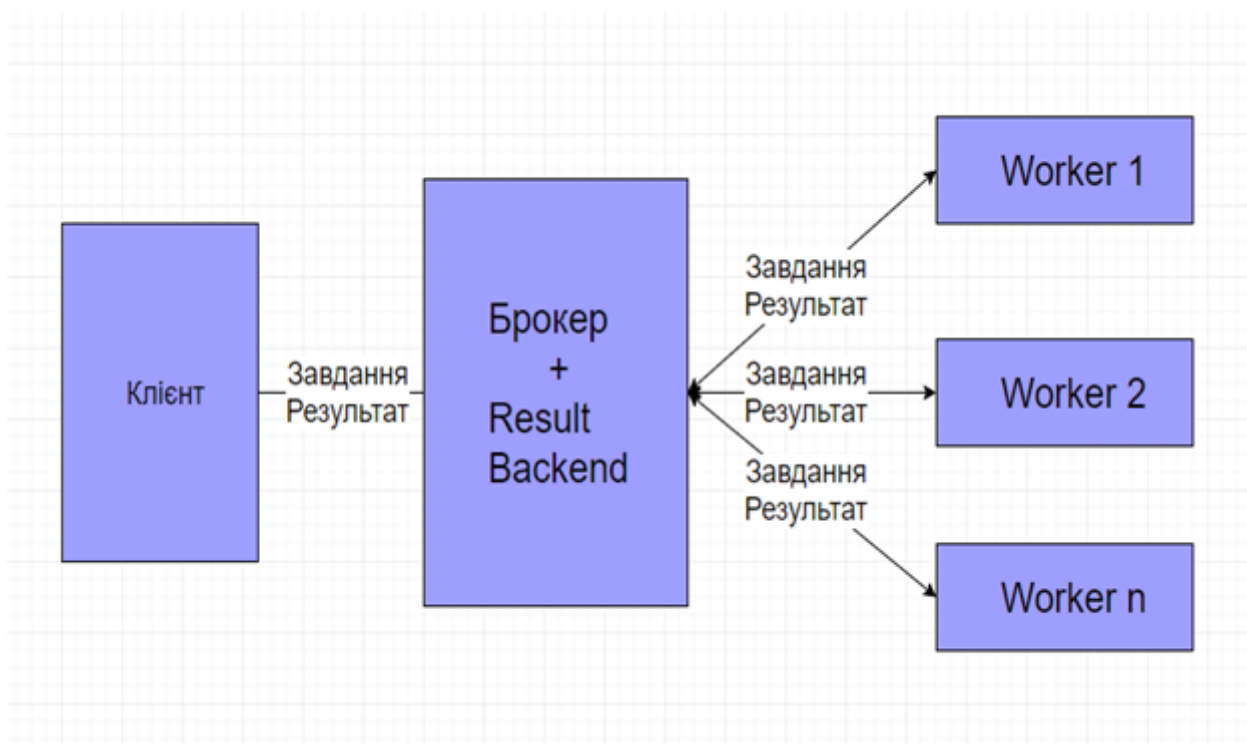


Рисунок 4.19 – Модель взаємодії хмарної САПР

Розроблена модель базується на використанні єдиної черги завдань, що знаходиться на “головному” вузлі кластера разом з веб-клієнтом. На всіх інших вузлах системи знаходяться “робітники”, кількість яких дорівнює кількості ядер вузла плюс один.

4.5.3 Особливості програмної реалізації

Одним з можливих способів реалізації моделі хмарної САПР є розробка web-серверу та менеджера черг на кластері з використанням фреймворку Celery для мови програмування Python. Celery – це асинхронна черга завдань з відкритим вихідним кодом або робоча черга, яка базується на переданому поширеному повідомленні. Хоча цей інструмент підтримує планування, його увага зосереджена на операціях в режимі реального часу.

Виконавчі одиниці, що називаються завданнями, виконуються одночасно на одному або декількох робочих вузлах, використовуючи багатопроцесорну, eventlet або gevent. Завдання можуть виконуватися асинхронно (у фоновому режимі) або синхронно (дочекатися, поки готовий).

Черги завдань використовуються як механізм для розподілу роботи по машинам. Вхідні данні черги завдань – це одиниця роботи, яка називається завданням (task). Виділені робочі процеси постійно відстежують черги завдань для виконання нових робіт.

Celery «спілкується» через повідомлення, зазвичай використовуючи брокера для посередництва між клієнтами та працівниками. Щоб розпочати завдання, клієнт додає повідомлення до черги, тоді брокер передає це повідомлення робітнику (worker).

Система Celery може складатися з декількох працівників та брокерів. Така архітектура дає можливість забезпечити високу доступність та горизонтальне масштабування.

При ініціалізації нового робітника вказується адреса черги та адреса, де буде зберігатися результат виконання завдання. У результаті ініціалізації працівника можлива у вигляді таких кодів мови програмування Python:

```
broker_url = 'redis://{host}:{port}/{db_number}'.format(  
    host=broker_host,  
    port=broker_port,  
    db_number=db_number  
)
```

```
app = Celery('executor', broker=broker_url)
```

```
app.conf.result_backend = broker_url
```

Через динамічний характеру Python особливо важким є виведення або перевірка типу об'єкта, що використовується. Для контролю типів даних у менеджері черг можливо використовувати анотації типів даних.

Анотації типу можуть бути вбудованими класами (включаючи ті, що визначені в стандартній бібліотеці або сторонніх модулях розширення), абстрактні базові класи, типи, доступні в модулі `types`, та визначені користувачем класи (у тому числі ті, що визначені в стандартній бібліотеці або сторонніх модулях).

Оскільки користувач передає вхідні дані на клієнт за допомогою REST API у форматі JSON, системі необхідно якось інтерпретувати та перевіряти їх. Наприклад, такі дві функції мають анотації для перевірки коректності параметрів, переданих до черги хмари:

```
def add(x: int, y: int) -> int:
    print("Task add called with args: {}, {}".format(x, y))
    return x + y
```

```
def bubble_sort(alist: list) -> list:
    for passnum in range(len(alist) - 1, 0, -1):
        for i in range(passnum):
            if alist[i] > alist[i + 1]:
                temp = alist[i]
                alist[i] = alist[i + 1]
                alist[i + 1] = temp
    return alist
```

Для реалізації клієнту менеджера черг у хмарі використовують, наприклад, фреймворки Django та Django Rest-framework. Django – це веб-фреймворк з відкритим вихідним кодом, написаний на Python, який базується на архітектурі model-view-template (MVT).

Незважаючи на наявність власної номенклатури, наприклад об'єкти для генерації HTTP відповідей називаються «view», Django можна розглядати, як MVC-фреймворк. Він складається з об'єктно-реляційного маппера (ORM), який є посередником між моделями даних та реляційною базою даних («Model»), системи для обробки HTTP-запитів «View» та диспетчера URL на основі регулярних виразів «Controller».

Клієнт отримує вхідні дані від користувача у форматі JSON. JSON – це формат даних, що не залежить від мови програмування. Якщо запитана користувачем функція існує у переліку доступних – відбувається перевірка відповідності типів вхідних даних вибраній задачі. Спочатку, за допомогою модуля `inspect`, зі стандартної бібліотеки, інспектується вибрана функція, з'ясовується перелік її вхідних параметрів та їх типи, чи мають аргументи функції значення за замовчуванням, якщо вони існують, то вони враховуються при створенні відповідного поля класу. Потім, на основі цих даних конструюється серіалайзер для перевірки вхідних даних та десеріалізації.

Серіалайзери дозволяють перетворити складні дані, такі як екземпляри моделей, у типи даних Python, які потім можна легко перетворити в JSON, XML або інші формати обміну даними. Серіалайзери також забезпечують десеріалізацію, що дозволяє відновлювати розібрані дані назад у складні типи після перевірки вхідних даних.

Якщо надані користувачем дані відповідають аргументам функції за своїми типами та кількістю – створюється нова задача, яка передається до черги хмари і буде виконана, коли на одному з вузлів кластера знайдеться вільний робітник. Створення нових задач реалізована у вигляді класу `CreateJobAPIView`:

```
class CreateJobAPIView(views.APIView):
    def get_task_serializer_class(self, task):
        return BaseTaskSerializer.for_func(task)

    def get_task_serializer(self, task, data):
        return self.get_task_serializer_class(task)(data=data)

    def post(self, *args, **kwargs):
        task = getattr(cluster_tasks, self.kwargs['task'], None)
        if task is None:
            raise Http404
        serializer = self.get_task_serializer(task, self.request.data)
        if serializer.is_valid(raise_exception=True):
            task = app.task(task).delay(**serializer.data)
            return Response({'task_id': task.id},
                            status=status.HTTP_201_CREATED)
```

Клієнту хмари видається `id`, за яким він може перевірити результат обчислення, коли він буде готовий:

```
class TaskResultAPIView(views.APIView):
    def get(self, *args, **kwargs):
```

```
res = AsyncResult(self.kwargs['id'], app=app)
if res.state == 'SUCCESS':
    return Response({
        'result': res.get()
    }, status=status.HTTP_200_OK)
else:
    return Response({
        'result': 'unknown'
    }, status=status.HTTP_200_OK)
```

ПІСЛЯМОВА

Основою сучасних САПР машинобудування є математичне моделювання, яке дозволяє віртуалізувати проведення експериментів. У таких галузях як, наприклад, аерокосмічна вартість випробувань зразків продукції є значною часткою у загальній собівартості продукції. Отже, розвиток наукових методів для САПР машинобудування дозволяє не тільки підвищити точність проектування, а й зменшити вартість продукції, що випускають підприємства.

Актуальною тенденцією розвитку сучасної комп'ютерної техніки та програмного забезпечення є поступовий перехід від традиційних програмних продуктів, які користувачі інсталиують на власні комп'ютери, до хмарних сервісів, орієнтованих на віддалене використання багатьма користувачами одночасно. Ця тенденція зумовлює необхідність розробки наукових методів, орієнтованих на використання у паралельних комп'ютерних системах.

Сучасні об'єкти машинобудування, зазвичай, мають складну форму. Їх поведінки описують системи диференціальних рівнянь, для розв'язання яких необхідно використовувати чисельні методи. Отже типова процедура дослідження об'єкта, що проектують, пов'язана з, по-перше, побудовою його геометричної моделі, по-друге, генерацією адекватної дискретної моделі, по-третє, проведення чисельного аналізу і обробкою його результатів. Протягом обробки результатів чисельного аналізу автоматично або напівавтоматично можуть бути змінені вхідні параметри попередніх етапів, що дозволить зробити таке моделювання адаптивним.

Типова задача побудови математичної моделі, що описує форму об'єкта машинобудування, як вихідні дані використовує креслення. На базі функціонального підходу можна побудувати модель об'єкта як єдину неявну функцію. Проте, у машинобудуванні зустрічаються об'єкти, форма яких описуються (повністю або частково) параметричними функціями. Запропонований математичний апарат дозволяє одночасно використовувати примітиви, межі яких визначено за допомогою параметричних і неявних функцій. Водночас моделі на базі параметричних функцій можуть виступати як операнди R-операцій. Отримані моделі можна розглядати як гібридні, що поєднують функціональне подання на базі неявних функцій і граничне подання на базі параметричних функцій.

Подання дискретної моделі множинами вузлів і елементів водночас зі зберіганням для кожного вузла разом з його координатами списку інцидентних

елементів дозволяє знизити порядок складності алгоритмів вставки та видалення вузлів. Спискові структури даних є більш ефективними при видаленні вузлів, а векторні – при безпечному додаванні (при вставці з перевіркою існування вузлів з однаковими координатами).

Проекційний метод генерації дискретних моделей меж об'єктів, форма яких задана функціонально, дозволяє будувати дво- та тривимірні дискретні моделі. Водночас щільність вузлів у підсумковій моделі залежить від щільності вузлів у вихідній. Проте, такий метод не враховує геометричних особливостей форми об'єкта, отже, повинна застосовуватися додаткова оптимізація.

Метод згладжування координат вузлів дискретних моделей меж об'єктів, форма яких задана функціонально, базується на мінімізації функціоналу, який враховує відстань від елементів дискретної моделі до поверхні та довжини їх сторін. Цей метод має локальний характер, тому потрібно його застосовувати ітераційно. За малої кількості ітерацій (менше чотирьох) можна отримати дискретні моделі форм об'єктів низької якості. Починаючи з четвертої ітерації, геометричні особливості форм об'єктів вже апроксимуються вузлами. Водночас виконання більше восьми ітерацій несуттєво змінює якість моделі. У результаті застосування цього методу мінімальні кути для трикутників несуттєво збільшуються, а у чотирикутниках можна отримати елементи, за формою близькі до трикутників, отже, результати загладжування потребують додаткової обробки топологічними перебудовами елементів.

Метод генерації дискретних моделей на базі елементів типу солід для форм об'єктів, заданих функціонально, дозволяє будувати дво- і тривимірні моделі як з використанням симплексів (трикутників або тетраедрів), так і з використанням топологічних кубів (чотирикутників або шестигранників). Цей метод також дозволяє використовувати спільний алгоритм для генерації дискретних моделей на базі різних форм скінченних елементів. Підсумкові дискретні моделі, зазвичай, потребують додаткової обробки.

Метод генерації адаптивних дискретних моделей форм об'єктів (adaptive meshing), заданих функціонально, який дозволяє будувати дво- і тривимірні дискретні моделі для виконання скінченно-елементного аналізу із потрібною точністю. Запропонований метод враховує результати скінченно-елементного аналізу, а не тільки інформацію про форму об'єкта. Як результат, кількість вузлів і елементів збільшується у тих областях, де найбільше змінюється потрібна досліднику величина. Метод не залежить від форми елементів. Він використовує таблиці шаблонів розбиття елементів. Отже, можна використовувати єдиний алгоритм для основних форм елементів.

Шаблони проектування САПР машинобудування як складної програмної системи є програмними абстракціями, які можна поєднати з іншими відомими

шаблонами для забезпечення більшої гнучкості та спрощення процесу супроводження програмного коду. Їх основною перевагою є можливість розробки САПР машинобудування без обмежень на вихідні коди.

Перевагою розроблених методів генерації дискретних моделей, скінченно-елементного аналізу та обробки чисельних результатів є їх узагальненість. Отже, можлива їх реалізація на базі шаблону проектування «Стратегія», що дозволяє підвищити загальну технологічність САПР машинобудування як програмного продукту.

Наближені аналітичні та аналітико-чисельні підходи, наприклад, гібридний ВКБ-Гальоркін метод, метод Рітца, метод малого параметра, дозволяють отримати оцінку напружено-деформованого стану елементів конструкцій у лінійній або нелінійній постановці як неперервні функції, що зумовлює можливість виконувати аналіз впливу різних параметрів конструкцій на стан без необхідності проводити додаткові обчислювальні експерименти. Введення програмної реалізації наведених методів у процесор хмарної САПР також сприяє додатковій верифікації скінченно-елементного аналізу виробів машинобудування базових форм.

Напрями подальших досліджень можуть бути пов'язані з використанням методів машинного навчання при створенні та оптимізації моделей поведінки об'єктів машинобудування, а також розробкою нейро-еволюційних методів прогнозування можливих зон руйнування в конструкціях з використанням чисельних методів як базових. Водночас реалізація нейро-еволюційних методів у хмарі буде потребувати розділення процедур навчання і прогнозування при балансуванні навантаження на комп'ютерну хмару.

ПЕРЕЛІК ПОСИЛАНЬ

1. Madenci E., Guven I. The finite element method and applications in engineering using ANSYS. New York: Springer, 2015. 656 P.
2. Efficient 3D Modeling of damage in composite materials / El-Sisi A. E.-D. A., H. M. El-Emam, H. A. Salim, H. E.-D. M. Sallam // Journal of Composite Materials. 2015. Vol. 49, No 7. P. 817–828. DOI: 10.1177/0021998314525983
3. Kiani S., Chan W. S., Haji-Sheikh A. Measurement of thermal conductivity of triaxial braided composites // Journal of Composite Materials. 2008. Vol. 42, No 12. P. 1159–1177. DOI: 10.1177/0021998308090652
4. Predicting Permeability based on flow simulations and textile modelling techniques: comparison with experimental values and verification of FlowTex Solver using Ansys CFX / E. E. Swery, R. Meier, St. V. Lomov [and others] // Journal of Composite Materials. 2016. Vol. 50, No 5. P. 601–614. DOI: 10.1177/0021998315579927
5. ANSYS LS-DYNA. URL: <http://www.ansys.com/Products/Structures/ANSYS-LS-DYNA> (дата звернення 25.05.2020).
6. ANSYS LS-DYNA. URL: <http://cae-expert.ru/product/ansys-ls-dyna> (дата звернення 25.05.2020).
7. MSC Nastran. Расчет и оптимизация конструкций. URL: <http://www.mscsoftware.ru/products/msc-nastran> (дата звернення 25.05.2020).
8. Turner T. L., Patel H. D. Analysis of SMA hybrid composite structures in MSC.Nastran and ABAQUS // Journal of Intelligent Material Systems and Structures. 2007. Vol. 18, Iss. 5. P. 435–447. DOI: 10.1177/1045389X06066699
9. Validation of a laminated beam model of LIPCA piezoelectric actuators / N. S. Goo, A. Haris, H. C. Park, K. J. Yoon // Journal of Intelligent Material Systems and Structures. 2005. Vol. 16, No 3. P. 189–195. DOI: 10.1177/1045389X05048043
10. A two-level optimization feature for the design of aerospace structures / E. Carrera, L. Mannella, G. Augello, N. Gualtieri // Journal of Aerospace Engineering. 2003. Vol. 217, No 4. P. 189–206. DOI: 10.1243/095441003769700753
11. MSC Nastran – multidisciplinary structural analysis. URL: <http://www.mscsoftware.com/product/msc-nastran> (дата звернення 25.05.2020).
12. Patran – complete FEA modeling solution. URL: <https://www.mscsoftware.com/product/patran> (дата звернення 25.05.2020).
13. Günel M., Kayran A. Non-linear progressive failure analysis of open-hole composite laminates under combined loading // Journal of Sandwich Structures and Materials. 2016. Vol. 15(3). P. 309–339. DOI: 10.1177/1099636213483651
14. Compression strength of stitched laminates after low-velocity impact / C. Xiaoquan, A. M. Al-Mansour, L. Zhengneng, K. Chenghe // Journal of Reinforced

Plastics and Composites. 2005. Vol. 24, No 9. P. 935–947. DOI: 10.1177/0731684405048193

15. Li D., Guo S., Xiang J. Modeling and nonlinear aeroelastic analysis of a wing with morphing trailing edge // Journal of Aerospace Engineering. 2013. Vol. 225, No 4. P. 619–631. DOI: 10.1177/0954410012438341

16. Dynamic Modeling and simulation of a two-stage series-parallel vibration isolation system / R. Guo, S.-Qi. Zhou, M.-Jia. Wang, S. Qui // Advances in Mechanical Engineering. 2016. Vol. 8(8). P. 1–19. DOI: 10.1177/1687814016660645

17. Puri G. Python Scripts for Abaqus. Learn by Example. USA: 2011. 725 P.

18. Meng Q., Jin Z., Fisher J. Comparison between FEBio and Abaqus for Biphasic Contact Problems // Journal of Engineering in Medicine. 2013. Volume 227, No 9. P. 1009–1019. DOI: 10.1177/0954411913483537

19. Lee Ch.-S., Lee J.-M. Anisotropic elasto-viscoplastic damage model for glass-fiber-reinforced polyurethane foam // Journal of Composite Materials. 2014. Vol. 48. P. 3367–3380. DOI: 10.1177/0021998313509863

20. Lawrence J., Li L. Finite element analysis of temperature distribution using ABAQUS for a laser-based tile grout sealing process // Journal of Engineering Manufacture. 2000. Vol. 214, No 6. P. 451–461. DOI: 10.1243/0954405001517766

21. Kettenis D. L. COSMOS: A simulation language for continuous, discrete and combined models // Simulation. 1992. Vol. 58. P. 32–41. DOI: 10.1177/003754979205800106

22. Mark W., Dukes-Schlossberg J. Cosmos: a system for supporting engineering negotiation // Concurrent Engineering. 1994. Volume 2. P. 173–182. DOI: 10.1177/1063293X9400200304

23. ЛИРА 9.4. Примеры расчета и проектирования : учебное пособие / В. Е. Боговис, Ю. В. Гензерский, Ю. Д. Гераймович [и др.]. Киев: Факт, 2008. 280 с.

24. ЛИРА 9.4. Руководство пользователя. / Е. Б. Стрелец-Стрелецкий, В. Е. Боговис, Ю. В. Гензерский [и др.] Основы. Киев: Факт, 2008. 164 с.

25. МОНОМАХ 4.2. Примеры расчета и проектирования. Учебное пособие / С. В. Юсипенко, Л. Г. Батрак, Д. А. Городецкий [и др.] Киев: Факт, 2007. 292 с.

26. Исследование задач механики деформируемого твердого тела в вычислительном комплексе «МИРЕЛА+» / Р. В. Киричевский, Е. С. Решевская, В. М. Тархова, Е. В. Прокопенко, В. В. Киричевский // Радіоелектроніка. Інформатика. Управління. 2006. № 2. С. 54–60.

27. Киричевский В. В. Система «КОДЕТОМ» для исследования высокоэластичных массивных конструкций на основе МКЭ // Комплексный расчет зданий и сооружений с применением ЭВМ. Киев: КИСИ, 1978. С. 142–148.

28. Киричевский В. В., Гребенюк С. Н., Мизерная Е. Л. Напряженно-деформированное состояние полого ортотропного цилиндра в условиях вязкоупругого деформирования // Нові матеріали і технології в металургії та машинобудуванні. 2005. № 2. С. 74–80.

29. Метод конечных элементов: теория, алгоритмы, реализация / В. А. Толок, В. В. Киричевский, С. И. Гоменюк [и др.] К.: Наукова думка, 2003. 316 с.

30. Моделирование напряженно-деформированного состояния композитов и конструкций на их основе / С. Н. Гребенюк, С. И. Гоменюк, Е. Л. Мизерная, В. В. Киричевский // *Радіоелектроніка. Інформатика. Управління*. 2006. № 2. С. 50–54.

31. Гоменюк С. И. Принципы визуализации результатов расчета трехмерных задач механики деформируемого твердого тела в инструментальной системе FORTU // *Вестник Херсонского государственного технического университета*. 2003. № 3(19). С. 102–105.

32. Гоменюк С. И. Объектно-ориентированные модели и методы анализа механических процессов // *Запорожский государственный университет*. Запорожье, 2004. С. 293–310.

33. Гоменюк С. И. Вычислительный комплекс для автоматизации проектирования сложных инженерных конструкций и сооружений // *Вестник Херсонского государственного технического университета*. 2004. № 1. С. 339–342.

34. Гоменюк С. І. Об'єктно-орієнтовані моделі і методи аналізу механічних процесів у системі автоматизації проектування: автореф. дис. на здобуття наук. ступеня д-ра техн. наук: [спец.] 05.13.12 “системи автоматизації проектувальних робіт” / Харківський національний університет радіоелектроніки. Харків, 2004. 34 с.

35. Гоменюк С. И., Мухин В. В., Морозов Д. Н. Инструментальный подход к построению универсальной САПР в машиностроении и строительстве FORTU-FEM // *Вестник Херсонского национального технического университета*. 2005. № 2. С. 107–113.

36. Гоменюк С. И., Гребенюк С. Н., Тархова В. М. Пространственно-временной конечный элемент для определения напряженно-деформированного состояния вязкоупругих материалов // *Пробл. обчисл. механіки і міцності конструкцій*. 2010. Вип. 14. С. 109–115.

37. Дырда В. И., Гребенюк С. Н., Гоменюк С. И. Аналитические и численные методы расчета резиновых деталей : монография. Днепропетровск; Запорожье, 2012. 370 с.

38. Рвачев В. Л., Шевченко А. Н. Проблемно-ориентированные языки и системы для инженерных расчетов. Київ: Техніка, 1988. 198 с.

39. Рвачев В. Л., Суворова И. Г., Шейко Т. И. Численная реализация задачи о течении вязкой жидкости методом R-функций в системе ПОЛЕ-3 // *Мат. методы и физ.-мех. поля*. 1991. № 33. С. 95–99.

40. Борисовская Ю. А., Гоменюк С. И. Современные САПР в аэрокосмической промышленности // *Вісник Запорізького національного університету. Фізико-математичні науки*. 2016. № 1. С. 20–31.

41. Борисовская Ю. А., Гоменюк С. И. Современные САПР в аэрокосмической промышленности // *Вісник Запорізького національного університету. Фізико-математичні науки*. 2016. № 1. С. 20–31.

42. Козлова О. С., Гоменюк С. И. Современные тенденции развития САПР в механике // *Вісник Запорізького національного університету. Фізико-математичні науки*. 2014. № 1. С. 38–51.

43. Agoston M. K. Computer graphics and geometric modelling:

implementation and algorithms. London : Springer-Verlag, 2005. 908 p.

44. Requicha A. A. G., Tilove R. B. Mathematical foundations of constructive solid geometry: general topology of closed regular sets [Technical Report TM-27]. Rochester, NY, USA: University of Rochester, 1978. 33 p.

45. Requicha A. A. G. Representations for rigid solids: theory, methods, and systems // Computing Surveys. 1980. Vol. 12, No 4. P. 437–464. DOI: 10.1145/356827.356833

46. Requicha A. A. G., Voelcker H. B. Solid modeling: a historical summary and contemporary assessment // IEEE Computer Graphics and Applications. 1982. Vol. 2, Iss. 2. P. 9–24. DOI: 10.1109/MCG.1982.1674149

47. Requicha A. A. G., Voelcker H. B. Solid modeling: current status and research directions // IEEE Computer Graphics and Applications. 1983. Vol. 3, Iss. 7. P. 25–37. DOI: 10.1109/MCG.1983.263271

48. Requicha A. A. G., Rossignac J. R. Solid modeling and beyond // IEEE Computer Graphics and Applications. 1992. Vol. 12, Iss. 5. P. 31–44. DOI: 10.1109/38.156011

49. Markowsky G., Wesley M. A. Fleshing out wire frames // IBM Journal of Research and Development. 1980. Vol. 24, Iss. 5. P. 582–597. DOI: 10.1147/rd.245.0582

50. Voelcker H. B., Requicha A. A. G. Geometric modeling of mechanical parts and processes // Computer. 1977. Vol. 10, Iss. 12. P. 48–57. DOI: 10.1109/C-M.1977.217601

51. March L., Steadman Ph. The geometry of environment: an introduction to spatial organization in design. Massachusetts: The MIT Press, 1974. 360 P.

52. Chen M., Tucker J. V. Constructive volume geometry // Computer Graphics Forum. 2000. Vol. 19, Iss. 4. P. 281–293. DOI: 10.1111/1467-8659.00464

53. Hable J., Rossignac J. CST: constructive solid trimming for rendering BReps and CSG // IEEE Transactions on Visualization and Computer Graphics. 2007. Vol. 13, Iss. 5. P. 1004–1014. DOI: 10.1109/TVCG.2007.70411

54. Rappoport A. Interactive boolean operations for conceptual design of 3-D solids // SIGGRAPH'97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques (Los Angeles, USA, 3–8 August 1997). New York, USA: ACM Press/Addison-Wesley Publishing Co, 1997. P. 269–278. DOI: 10.1145/258734.258865

55. Rossignac J. R. Constraints in constructive solid geometry. Interactive 3D graphics: Proceedings of the 1986 workshop on, Chapel Hill, NC, USA, 23–24 October 1986. New York: ACM, 1987. P. 93–110. DOI: 10.1145/319120.319129

56. Krysl P., Ortiz M. Extraction of boundary representation from surface triangulations // International Journal for Numerical Methods in Engineering. 2001. Vol. 50, Iss. 7. P. 1737–1758. DOI: 10.1002/nme.94

57. Owen S. J., White D. R. Mesh-based geometry // International Journal for Numerical Methods in Engineering. 2003. Vol. 58, Iss. 2. P. 375–395. DOI: 10.1002/nme.773

58. Quadros W. R., Owen S. J. Defeaturing CAD models using a geometry-based size field and facet-based reduction operators // Engineering with Computers.

2012. Vol. 28, Iss. 3. P. 211–224. DOI: 10.1007/s00366-011-0252-8

59. Stroud I. Boundary representation modelling techniques. London: Springer, 2006. 808 P. ISSN: 1846283124.

60. Vaněček G. Brep-index: a multidimensional space partitioning tree // International Journal of Computational Geometry & Applications. 1991. Vol. 1, Iss. 3. P. 243–261. DOI: 10.1142/S0218195991000189

61. Рвачев В. Л., Ющенко Е. Л. Некоторые вопросы аналитического описания геометрических объектов сложной логической системы. Киев: Общество «Знание» УССР, 1965. 53 с.

62. Рвачев В. Л. Геометрические приложения алгебры логики. Київ: Техніка, 1967. 212 с.

63. Рвачев В. Л. Теория R-функций и некоторые ее приложения. Київ: Наукова Думка, 1982. 552 с.

64. Рвачев В. Л., Шапиро В., Шейко Т. И. Метод R-функций (RFM) в краевых задачах с геометрической и физической симметрией // Математичні методи та фізико-механічні поля. 1998. № 1. С. 146–159.

65. Рвачев В. Л., Шапиро В., Шейко Т. И. Применение метода R-функций к построению уравнений локусов, обладающих симметрией // Электромагнитные волны и электронные системы. 1999. № 4. С. 4–20.

66. Рвачев В. Л., Уваров Р. А., Шейко Т. И. Построение уравнений локусов в 3D с помощью R-функций // Радиоэлектроника и информатика. 2001. № 1. С. 158–164.

67. Рвачев В. Л., Шейко Т. И. Введение в теорию R-функций // Проблемы машиностроения. 2001. Т. 4. № 1–2. С. 46–58.

68. Рвачев В. Л., Уваров Р. А., Шейко Т. И. Интерлокационные формулы при моделировании задач Плато и Софи Жермен // Радиоэлектроника и информатика. 2002. № 3. С. 126–128.

69. Максименко-Шейко К. В., Мацевитый А. М., Шейко Т. И. Автоматизация построения уравнений геометрических объектов в методе R-функций // Кибернетика и системный анализ. 2006. № 2. С. 148–157.

70. Максименко-Шейко К. В., Шейко Т. И. R-функции в математическом моделировании геометрических объектов, обладающих симметрией // Кибернетика и системный анализ. 2008. № 6. С. 75–83.

71. Максименко-Шейко К. В. Метод R-функций в математическом моделировании теплообмена при движении жидкости по цилиндрическим каналам с пристеночными винтовыми вставками // Вісник Харківського національного університету імені В. Н. Каразіна. Математичне моделювання. Інформаційні технології. Автоматизовані системи управління. 2009. № 863(12). С. 180–190.

72. Максименко-Шейко К. В. Нормальные и нормализованные уравнения геометрических объектов в методе R-функций // Вісник Харківського національного університету імені В. Н. Каразіна. Математичне моделювання. Інформаційні технології. Автоматизовані системи управління. 2009. № 847(11). С. 251–265.

73. Максименко-Шейко К. В. R-функции в математическом моделировании геометрических объектов и физических полей. Харьков: ИПМаш НАН Украины, 2009. 306 с.

74. Максименко-Шейко К. В., Шейко Т. И. R-функции в математическом моделировании геометрических объектов в 3D по информации в 2D // Вісник Запорізького національного університету: Збірник наукових праць. Фізико-математичні науки. 2010. № 1. С. 98–104.
75. Максименко-Шейко К. В., Шейко Т. И. Математическое моделирование тепловых режимов радиоэлектронных плат с размещением источников по схеме «ковёр Серпинского» // Математичні методи та фізико-механічні поля. 2012. Т. 55. № 3. С. 170–177.
76. Максименко-Шейко К. В., Шейко Т. И. Математическое моделирование геометрических фракталов с помощью R-функций // Кибернетика и системный анализ. 2012. Т. 48. № 4. С. 155–162.
77. Blinn J. F. A generalization of algebraic surface drawing // ACM Transactions on Graphics. 1982. Vol. 1, Iss. 3. P. 235–256. DOI: 10.1145/357306.357310
78. Pasko A., Adzhiev V., Sourin A., Savchenko V. Function representation in geometric modeling: concepts, implementation and applications // The Visual Computer. Vol. 11, No 8. P. 429–446.
79. Pasko A., Adzhiev V. Function-based shape modeling: mathematical framework and specialized language // lecture notes in computer science. 2004. Vol. 2930. P. 132–160.
80. Shapiro V. Semi-analytic geometry with R-functions // Acta Numerica. 2007. Vol. 16. P. 239–303. DOI: 10.1017/S096249290631001X
81. Lorensen W. E., Cline H. E. Marching cubes: a high resolution 3D surface construction algorithm // Computer Graphics. 1987. Vol. 21, Iss. 4. P. 163–169.
82. Web-based shape modeling with HyperFun / R. Cartwright, V. Adzhiev, A. A. Pasko, Y. Goto, T. L. Kunii // IEEE Computer Graphics and Applications. 2005. Vol. 25, Iss. 2. P. 60–69.
83. Paoluzzi A. Geometric programming for computer aided design. Wiley, 2003. 800 p.
84. Paoluzzi A. Informatica grafica en CAD. Hoepli Informatica, 2003. 424 p.
85. Yates D. N., Sable W. W., Vinson T. J. The DAISY code // Numerical and Computer Methods in Structural Mechanics. New-York, 1973. P. 175–210.
86. Geuzaine C., Remacle J.-F. Gmsh: a three-dimensional finite element mesh generator with built-in pre- and post-processing facilities // International Journal for Numerical Methods in Engineering. 2009. Vol. 79, Iss. 11. P. 1309–1331. DOI: 10.1002/nme.2579
87. Бувайло Д. П., Гоменюк С. І., Толок В. О. FORTU – мова опису схем розв’язання задач математичної фізики // Вісник Запорізького державного університету. 2000. № 1. С. 19–25.
88. Buhmann M. D. Radial basis functions: theory and implementations. Cambridge: Cambridge University Press, 2004. 271 P.
89. Liu G. R., Gu Y. T. An introduction to meshfree methods and their programming. Dordrecht, The Netherlands: Springer, 2005. 496 P.
90. Liu G. R. Mesh free methods: moving beyond the finite element method. Boca Raton, London, New York, Washington, D.C.: CRC Press, 2003. 693 P.

91. Rahman S., Rao B. N. An element-free galerkin method for probabilistic mechanics and reliability // *International Journal of Solids and Structures*. 2001. Vol. 38. P. 9313–9330. DOI: 10.1016/S0020-7683(01)00193-7
92. Zienkiewicz O. C., Taylor R. L., Zhu J. Z. The finite element method: its basis and fundamentals, Seventh Edition. London: Butterworth-Heinemann, 2013. 756 P.
93. Zienkiewicz O. C., Taylor R. L.. The finite element method, Fifth Edition, Volume 2: Solid Mechanics. London: Butterworth-Heinemann, 2000. 460 P.
94. Zienkiewicz O. C., Taylor R. L., Nithiarasu P. The finite element method for fluid dynamics, seventh edition. London: Butterworth-Heinemann, 2013. 584 P.
95. Kumar Pr., Saxena A. On embedded recursive boundary smoothing in topology optimization with polygonal mesh and negative masks // *Conference on Machines and Mechanism: Proceedings of the 1st International and 16th National Conference (IIT Roorkee, India, 18–20 December 2013)*. 2013. P. 568–576.
96. Kumar Pr., Saxena A. On topology optimization with embedded boundary resolution and smoothing // *Structural and Multidisciplinary Optimization*. 2015. Vol. 52, Iss. 6. P. 1135–1159. DOI: 10.1007/s00158-015-1272-6
97. Langelaar M. The use of convex uniform honeycomb tessellations in structural topology optimization // *Structural and Multidisciplinary Optimization: Proceedings of the 7th world congress (Seoul, South Korea, 21–25 May 2007)*. Seoul: 2007. DOI: 10.1007/s00158-008-0261-4
98. MacDonald A. E., Middlecoff J., Henderson T. A general method for modeling on irregular grids // *The International Journal of High Performance Computing Applications*. 2010. Vol. 25, Iss. 4. P. 392–403. DOI: 10.1177/1094342010385019
99. Saxena A. A material-mask overlay strategy for continuum topology optimization of compliant mechanisms using honeycomb discretization // *ASME Journal of Mechanism Design*. 2009. Vol. 130, Iss. 8. P. 082304:1–9. DOI: 10.1115/1.2936891
100. Saxena A., Jain C. An improved material-mask overlay strategy for topology optimization of structures and compliant mechanisms // *ASME Journal of Mechanism Design*. 2010. Vol. 132, Iss. 6. P. 061006:1–10. DOI: 10.1115/1.4001530
101. Saxena A. An adaptive material mask overlay method: modifications and investigations on binary, well connected robust compliant continua // *ASME Journal of Mechanism Design*. 2009. Vol. 133, Iss. 4. P. 041004:1–11. DOI: 10.1115/1.4003804
102. Saxena A. Topology optimization with negative masks using gradient search // *Structural and Multidisciplinary Optimization*. 2011. Vol. 44, Iss. 5. P. 629–649. DOI: 10.1007/s00158-011-0649-4
103. Saxena A. Combined gradient-stochastic optimization with negative circular masks for large deformation topologies // *International Journal for Numerical Methods in Engineering*. 2013. Vol. 93, Iss. 6. P. 635–663. DOI: 10.1002/nme.4401
104. Saxena R., Saxena A. On honeycomb representation and SIGMOID material assignment in optimal topology synthesis of compliant mechanisms // *Finite Elements in Analysis and Design*. 2007. Vol. 43, Iss. 14. P. 1082–1098. DOI: 10.1016/j.finel.2007.08.004

105. Talischi C., Paulino G. H., Le Ch. H. Honeycomb Wachspress finite elements for structural topology optimization // Structural and Multidisciplinary Optimization. 2009. Vol. 37, Iss. 6. P. 569–583. DOI: 10.1007/s00158-008-0261-4
106. Talischi C., Paulino G. H., Pereira A. PolyMesher: a general-purpose mesh generator for polygonal elements written in Matlab // Structural and Multidisciplinary Optimization. 2012. Vol. 45, Iss. 3. P. 309–328. DOI: 10.1007/s00158-011-0706-z
107. Talischi C., Pereira A., Paulino G. H. Polygonal finite elements for incompressible fluid flow // international journal for numerical methods in fluids. 2014. Vol. 74. P. 134–151. DOI: 10.1002/fld.3843
108. Heikes R., Randall D. Numerical integration of the shallow-water equations on a twisted icosahedral grid. Part I: basic design and results of tests // monthly weather review. 1995. Vol. 123. P. 1862–1880.
109. Heikes R., Randall D. Numerical integration of the shallow-water equations on a twisted icosahedral grid. Part II: a detailed description of the grid and an analysis of numerical accuracy // monthly weather review. 1995. Vol. 123. P. 1881–1887.
110. The non-hydrostatic icosahedral atmospheric model: description and development / m. Satoh, H. Tomita, H. Yashiro [and others] // Progress in Earth and Planetary Science. 2014. Vol. 18, Iss. 1. P. 1–32. DOI: 10.1186/s40645-014-0018-1
111. Stuhne G. R., Peltier W. R. New icosahedral grid-point discretizations of the shallow water equations on the sphere // journal of computational physics. 1999. Vol. 148. P. 23–58.
112. Wang N., Lee J.-L. Geometric properties of the icosahedral-hexagonal grid on the two-sphere // siam journal on scientific computing. 2011. Vol. 33, No 5. P. 2536–2559. DOI: 10.1137/090761355
113. Thompson J. F., Sony Bh., Weatherill N. Handbook of grid generation. New York: CRC Press, 1999. 1136 p.
114. Gordon W. N., Hall C. A. Construction of curvilinear coordinate systems and application to mesh generation // international journal for numerical methods in engineering. 1973. Vol. 7, Iss. 4. P. 461–477. DOI: 10.1002/nme.1620070405
115. Cook W. A. Body oriented (natural) coordinates for generating three dimensional meshes // international journal for numerical methods in engineering. 1974. Vol. 8, Iss. 1. P. 27–43. DOI: 10.1002/nme.1620080104
116. Haber R., Shephard M. S., Abel J. F. A general two-dimensional, graphical finite element preprocessor utilizing discrete transfinite mapping // international journal for numerical methods in engineering. 1981. Vol. 17, Iss. 7. P. 1015–1044. DOI: 10.1002/nme.1620170706
117. Haber R., Abel J. F. Discrete transfinite mappings for description and meshing of three-dimensional surfaces using interactive computer graphics // international journal for numerical methods in engineering. 1982. Vol. 18, Iss. 1. P. 41–66. DOI: 10.1002/nme.1620180105
118. Eiseman P. R. A multi-surface method of coordinate generation // journal of computational physics. 1979. Vol. 33, Iss. 1. P. 118–150.
119. Флетчер К. Вычислительные методы в динамике жидкостей. Москва: Мир, 1991. Т. 2: Методы расчета различных течений. 552 с.

120. Floater M. S. Parametrization and smooth approximation of surface triangulations // computer aided geometric design. 1997. Vol. 14, Iss. 3. P. 231–250. DOI: 10.1016/S0167-8396(96)00031-3

121. Floater M. S. Mean value coordinates // computer aided geometric design. 2003. Vol. 20, Iss. 1. P. 19–27. DOI: 10.1016/S0167-8396(03)00002-5

122. Floater M. S., Hormann K., Kós G. A general construction of barycentric coordinates over convex polygons // advances in computational mathematics. 2006. Vol. 24, Iss. 1. P. 311–331. DOI: 10.1007/s10444-004-7611-6

123. Meyer M., Barr A., Lee H. Generalized barycentric coordinates on irregular polygons // journal of graphics tools. 2002. Vol. 7, Iss. 1. P. 13–22. DOI: 10.1080/10867651.2002.10487551

124. Warren J. Barycentric coordinates for convex polytopes // advances in computational mathematics. 1996. Vol. 6, Iss. 1. P. 97–108. DOI: 10.1007/BF02127699

125. Lehtimäki R. An algebraic boundary orthogonalization procedure for structured grids // International Journal for Numerical Methods in Fluids. 2000. Vol. 32, Iss. 5. P. 605–618. DOI: 10.1002/(SICI)1097-0363(20000315)32:5<605::AID-FLD986>3.0.CO;2-5

126. Yerry M. A., Shephard M. S. A modified quadtree approach to finite element mesh generation // IEEE Computer Graphics and Applications. 1983. Vol. 3, Iss. 1. P. 39–46.

127. Robust, geometrically based, automatic two-dimensional mesh generation / P. L. Baehmann, S. L. Wittchen, M. S. Shephard [and others] // International Journal for Numerical Methods in Engineering. 1987. Vol. 24, Iss. 6. P. 1043–1078. DOI: 10.1002/nme.1620240603

128. Liang X., Ebeida M. S., Zhang Y. Guaranteed-quality all-quadrilateral mesh generation with feature preservation // Computer Methods in Applied Mechanics and Engineering. 2010. Vol. 199. P. 2072–2083. DOI: 10.1007/978-3-642-04319-2_4

129. Quadtree grid generation: information handling, boundary fitting and CFD applications / K. F. C. Yiu, D. M. Greaves, S. Cruz [and others] // Computers & Fluids. 1996. Vol. 25, Iss. 8. P. 759–769.

130. Ito Y., Shih A. M., Soni B. K. Octree-based reasonable-quality hexahedral mesh generation using a new set of refinement templates // International Journal for Numerical Methods in Engineering. 2009. Vol. 77, Iss. 13. P. 1809–1833. DOI: 10.1002/nme.2470

131. Kolšek T., Šubelj M., Duhovnik J. Generation of block-structured grids in complex computational domains using templates // Finite Elements in Analysis and Design. 2003. Vol. 39. P. 1139–1154.

132. Guoy D., Erickson J. Automatic blocking scheme for structured meshing in 2D multiphase flow simulation // International Meshing Roundtable: Proceedings of the 13th International Conference (Willimasburg, Virginia, USA, 19–22 September 2004). Sandia: Sandia National Laboratories, 2004. P. 121–132.

133. Quadros W. R., Ramaswami K., Prinz F. B. LayTracks: a new approach to automated geometry adaptive quadrilateral mesh generation using medial axis

transform // International Journal for Numerical Methods in Engineering. 2004. Vol. 61, Iss. 2. P. 209–237. DOI: 10.1002/nme.1063

134. Tam T. K. H., Armstrong C. G. 2D finite element mesh generation by medial axis subdivision // Advances in Engineering Software and Workstations. 1991. Vol. 13, No 5-6. P. 313–324.

135. Taghavi R. Automatic block decomposition using fuzzy logic analysis // International Meshing Roundtable: Proceedings of the 9th International Conference, New Orleans, Louisiana, USA, 2–5 October 2000. Sandia: Sandia National Laboratories, 2000. P. 187–192.

136. Takahashi H., Shimizu H. A general purpose automatic mesh generation using shape recognition technique // ASME Computers in Engineering. 1991. Vol. 1. P. 519–526.

137. Takahashi H., Shimizu H., Moriyama H. A three-dimensional automatic mesh generation system using shape recognition technique // Transactions of the Japan Society of Mechanical Engineers. 1993. Vol. 59, No 560, Series A. P. 1161–1167.

138. Müller-Hannemann M. High quality quadrilateral surface meshing without template restrictions: a new approach based on network flow techniques // The International Journal of Computational Geometry and Applications. Vol. 10, Iss. 3. P. 285–307.

139. Jablonowski C., Oehmke R. C., Stout Q. F. Block-structured adaptive meshes and reduced grids for atmospheric general circulation models // Philosophical Transactions of the Royal Society A. 2009. Volume 367. P. 4497–4522.

140. Zheng J. Z. X., Groth C. P. T. Block-based adaptive mesh refinement finite-volume scheme for hybrid multi-block meshes // Computational Fluid Dynamics (ICCFD7): Proceedings of the 7th International Conference (Big Island, Hawaii, 9–13 July 2012). Hawaii: 2012. P. 1–19.

141. Ahusborde E., Glockner S. A 2D block-structured mesh partitioner for accurate flow simulations on non-rectangular geometries // Computers & Fluids. 2011. Vol. 43. P. 2–13. DOI: 10.1016/j.compfluid.2010.07.009

142. Applying Parmetis to structured remeshing for industrial CFD applications / S. Laflamme, J. Dompierre, F. Guibault, R. Roy // The International Journal of High Performance Computing Applications. 2003. Vol. 17, No 1. P. 63–76. DOI: 10.1177/1094342003017001006

143. Lee H., Lee K. O. Multi-layered mesh generation and user subroutine development based on an Eulerian–Eulerian approach for soot filtration visualization in a three-dimensional particulate filter model // International Journal of Engine Research. 2014. Vol. 15, Iss. 8. P. 980–992. DOI: 10.1177/1468087414543707

144. Делоне Б. Н. О пустой сфере // Известия АН СССР. ОМЕМ. 1934. № 4. С. 793–800.

145. Watson D. F. Computing the N-dimensional Delaunay tessellation with applications to Voronoi polytopes // The Computer Journal. 1981. Vol. 24, Iss. 2. P. 167–172.

146. Watson D. F. Contouring: A guide to the analysis and display of spatial data. Oxford: Pergamon Press, 1992. 321 p.

147. Скворцов А. В. Триангуляция Делоне и её применение. Томск :

Издательство Томского университета, 2002. 128 с.

148. Скворцов А. В. Обзор алгоритмов построения триангуляции Делоне // Вычислительные методы и программирование. 2002. № 3. С. 14–39.

149. Скворцов А. В., Мирза Н. С. Алгоритмы построения и анализа триангуляции. Томск : Издательство Томского университета, 2006. 168 с.

150. Su P., Drysdale R. L. S. A comparison of sequential Delaunay triangulation algorithms // Computational Geometry. 1997. Vol. 7, Iss. 5-6. P. 361–385.

151. Joe B. Delaunay Triangular Meshes in Convex Polygons // SIAM Journal on Scientific and Statistical Computing. 1986. Vol. 7, Iss. 2. P. 514–539. DOI: 10.1137/0907035

152. Joe B. Three-dimensional triangulations from local transformations // SIAM Journal on Scientific and Statistical Computing. 1989. Vol. 10. P. 718–741. DOI: 10.1137/0910044

153. Joe B. Construction of three-dimensional triangulations using local transformations // Computer Aided Geometric Design. 1991. Vol. 8. P. 123–142.

154. Joe B. Construction of k-dimensional delaunay triangulations using local transformations // SIAM Journal on Scientific Computing. 1993. Vol. 14, Iss. 6. P. 1415–1436. DOI: 10.1137/0914083

155. Joe B. Construction of three-dimensional improved-quality triangulations using local transformations // SIAM Journal on Scientific Computing. 1995. Vol. 16, Iss. 6. P. 1292–1307. DOI: 10.1137/0916075

156. Shewchuk J. R. Robust adaptive floating-point geometric predicates. Computational Geometry: Proceedings of the 12th Annual ACM Symposium (Philadelphia, Pennsylvania, 24–26 May 1996). Philadelphia: ACM, 1996. P. 141–150.

157. Bern M., Eppstein D. Mesh generation and optimal triangulation // Computing in Euclidean Geometry. 1992. No 1. P. 23–90.

158. Bern M., Eppstein D., Gilbert J. Provably good mesh generation // Journal of Computer and System Sciences. 1994. Vol. 48, Iss. 3. P. 384–409.

159. Bern M., Dobkin D., Eppstein D. Triangulating polygons without large angles // International Journal of Computational Geometry & Applications. 1995. Vol. 5. P. 171–192.

160. Chew L. P. Constrained Delaunay triangulations // Algorithmica. 1989. Vol. 4, Iss. 1. P. 97–108.

161. Chew L. P. Guaranteed-quality triangular meshes. Technical Report TR-89-983. Ithaca, NY: Cornell University, 1989. 20 p.

162. Ruppert J. A Delaunay refinement algorithm for quality 2-dimensional mesh generation // Journal of Algorithms. 1995. Vol. 18, Iss. 3. P. 548–585

163. Tournois J., Alliez P., Devillers O. Interleaving Delaunay refinement and optimization for 2D triangle mesh generation // International Meshing Roundtable: Proceedings of the 16th International Conference (Washington, 14–17 October 2007). Washington: Springer, 2008. P. 83–101.

164. CGALmesh: a generic framework for Delaunay mesh generation / C. Jamin, P. Alliez, M. Yvinec, J.-D. Boissonnat // ACM Transactions on Mathematical Software. 2014. Vol. 1, No 1. P. 1–26.

165. Cheng S.-W., Dey T. K., Shewchuk J. R. Delaunay mesh generation. Boca

Raton, FL: CRC Press, 2013. 387 p.

166. Cohel-Steiner D., de Verdière É. C., Yvinec M. Conforming Delaunay triangulations in 3D // *Computational Geometry: Theory and Applications*. 2004. Vol. 28. P. 217–233.

167. Frey W. H. Selective refinement: a new strategy for automatic node placement in graded triangular meshes // *International Journal for Numerical Methods in Engineering*. 1987. Vol. 24, Iss. 11. P. 2183–2200. DOI: 10.1002/nme.1620241111

168. Du Q., Wang D. Tetrahedral mesh generation and optimization based on centroidal Voronoi tessellations // *International Journal for Numerical Methods in Engineering*. 2003. Vol. 56, Iss. 9. P. 1355–1373. DOI: 10.1002/nme.616

169. Du Q., Wang D. Recent progress in robust and quality Delaunay mesh generation // *Journal of Computational and Applied Mathematics*. 2006. Vol. 195, Iss. 1-2. P. 8–23.

170. Hazelwood C. Approximating constrained tetrahedrizations // *Computer Aided Geometric Design*. 1993. Vol. 10. P. 67–87.

171. Nigel P. W. Delaunay triangulation in computational fluid dynamics // *Computers and Mathematics with Applications*. 1992. Vol. 24, No 5/6. P. 129–150.

172. Nigel P. W., Hassan O. Efficient three-dimensional grid generation using the Delaunay triangulation // *Computational Fluid Dynamics: Proceedings of the First European Conference (Brussels, 7–11 Belgium September 1992)*. Amsterdam, New York: Elsevier, 1992. P. 961–968.

173. George P. L., Hecht F., Saltel E. Automatic mesh generator with specified boundary // *Computer Methods in Applied Mechanics and Engineering*. 1991. Vol. 92, Iss. 3. P. 269–288.

174. Barth T. J., Jespersen D. C. The design and application of upwind schemes on unstructured meshes // *The 27th Aerospace Sciences Meeting: Proceedings of Multidisciplinary Meeting (Reno, NV, USA, 9–2 January 1989)*. Reno: AIAA, 1989. AIAA PAPER 89-0366. 13 p. DOI: 10.2514/6.1989-366

175. Frey P., Borouchaki H., Geouge P.-L. Delaunay tetrahedralization using an advancing-front approach // *International Meshing Roundtable: Proceedings of the 5th International Conference (Pittsburgh, Pennsylvania, U.S.A., 10–11 October 1996)*. Sandia: Sandia National Laboratories, 1996. P. 31–48.

176. Frey P., Borouchaki H., Geouge P.-L. 3D Delaunay mesh generation coupled with an advancing-front approach // *Computer methods in applied mechanics and engineering*. 1998. Vol. 157, Iss. 1. P. 115–131.

177. Kucwaj J., Borowik B. An algorithm of combining of advancing front and Delaunay triangulation to 3-D domain tessellation // *Annales UMCS Informatica AI IX*. 2009. No 1. P. 15–26.

178. Löhner R. Generation of three-dimensional unstructured grids by the advancing front method // *International Journal for Numerical Methods in Fluids*. 1988. Vol. 8. P. 1135–1149.

179. Marcum D. L., Weatherill N. P. Unstructured grid generation using iterative point insertion and local reconnection // *AIAA Journal*. 1995. Vol. 33, No 9. P. 1619–1625.

180. Mavriplis D. J. An advancing front Delaunay triangulation algorithm designed

for robustness. *Journal of Computational Physics*. 1995. Vol. 117, Iss. 1. P. 90–101.

181. Müller J.-D., Poe P. L., Deconinck H. A frontal approach for internal node generation in Delaunay triangulations // *International Journal for Numerical Methods in Fluids*. 1993. Vol. 17. P. 241–255.

182. Silva C. T., Mitchell J. S. B. Greedy Cuts: An Advancing Front Terrain Triangulation Algorithm // *Advances in Geographic Information Systems: Proceedings of The 6th ACM International Symposium* (Washington, 2–7 November 1998). Washington: ACM, 1998. P. 137–144. DOI: 10.1145/288692.288717

183. Галанин М. П., Щеглов И. А. Разработка и реализация алгоритмов трехмерной триангуляции сложных пространственных областей: итерационные методы. Москва: ИПМ им М. В. Келдыша РАН, 2006. № 9. 32 с. (Препринт / РАН, ИПМ им. М. В. Келдыша ; 06-01-00421).

184. Baker B. S., Grosse E., Rafferty C. S. Nonobtuse triangulation of polygons // *Discrete and Computational Geometry*. 1988. Vol. 3, Iss. 2. P. 147–168.

185. Mitchell S. A., Vavasis S. A. Quality Mesh Generation in Three Dimensions. *Computational Geometry: Proceedings of the Eighth Annual Symposium* (Berlin, Germany, 10–12 June 1992). New York: Cornell University, 1992. P. 92–104.

186. Mitchell S. A., Vavasis S. A. Quality Mesh Generation in Higher Dimensions // *SIAM Journal on Computing*. 2000. Vol. 29, Iss. 4. P. 1334–1370.

187. Samet H. The Quadtree and Related Hierarchical Data Structures // *ACM Computing Surveys*. 1984. Vol. 16, Iss. 2. P. 187–260.

188. Yerry M. A., Shephard M. S. Automatic three-dimensional mesh generation by the modified-octree technique // *International Journal for Numerical Methods in Engineering*. 1984. Vol. 20, Iss. 11. P. 1965–1990. DOI: 10.1002/nme.1620201103

189. Neugebauer F., Diekmann R. Improved Mesh Generation: Not Simple but Good // *International Meshing Roundtable: Proceedings of the 5th International Conference* (Pittsburgh, Pennsylvania, U.S.A., 10–11 October 1996). Sandia: Sandia National Laboratories, 1996. P. 257–272.

190. A new approach to the development of automatic quadrilateral mesh generation / J. Z. Zhu, O. C. Zienkiewicz, E. Hinton, J. Wu // *International Journal For Numerical Methods in Engineering*. 1991. Vol. 32, Iss. 4. P. 849–866. DOI: 10.1002/nme.1620320411

191. A new automated scheme of quadrilateral mesh generation for randomly distributed line constraints / C. Park, J.-S. Noh, I.-S. Jang, J. M. Kang // *Computer-Aided Design*. 2007. Vol. 39. P. 258–267. DOI: 10.1016/j.cad.2006.12.002

192. Blacker T. D., Stephenson M. B. Paving: A new approach to automated quadrilateral mesh generation // *International Journal For Numerical Methods in Engineering*. 1991. Vol. 32, Iss. 4. P. 811–847. DOI: 10.1002/nme.1620320410

193. Rees M. Combining quadrilateral and triangular meshing using the advancing front approach // *International Meshing Roundtable: Proceedings of the 6th International Conference* (Park City, Utah, USA, 13–15 October 1997). Sandia: Sandia National Laboratories, 1997. P. 337–348.

194. Blacker T. D., Meyers R. J. Seams and wedges in plastering: a 3D

hexahedral mesh generation algorithm // Engineering with Computers. 1993. Vol. 9, Iss. 2. P. 83–93.

195. Shimada K., Liao J.-H., Itoh T. Quadrilateral meshing with directionality control though the packing of square control // International Meshing Roundtable: Proceedings of the 7th International Conference (Dearborn, Michigan, U.S.A., 26–28 October 1998). Sandia: Sandia National Laboratories, 1998. P. 61–76.

196. Zhao Y., Yu B. A modified paving algorithm for quadrilateral mesh generation // Kung Cheng Je Wu Li Hsueh Pao/Journal of Engineering Thermophysics. 2013. Vol. 34, Iss. 4. P. 728–732.

197. Staten M. L., Owen S. J., Blacker T. D. Unconstrained paving & plastering: a new idea for all hexahedral mesh generation // International Meshing Roundtable: Proceedings of the 14th International Conference (San Diego, California, USA, 11–14 September 2005). Sandia National Laboratories: Springer-Verlag, 2005. P. 399–416. DOI: 10.1007/3-540-29090-7_24

198. Staten M. L., Owen S. J., Blacker T. D. Unconstrained plastering – Hexahedral mesh generation via advancing-front geometry decomposition // International Journal for Numerical Methods in Engineering. 2010. Vol. 81, Iss. 2. P. 135–171. DOI: 10.1002/nme.2679

199. Сковпень А. В. Усовершенствованный алгоритм построения нерегулярных четырехугольных сеток. Снежинск: РФ ЯЦ-ВНИИТФ, 2004. 62 с.

200. Сковпень А. В. Усовершенствованный алгоритм построения нерегулярных четырехугольных сеток // Журнал вычислительной математики и математической физики. 2005. Т. 45. № 8. С. 1506–1528.

201. Тимирязев В. А., Белянкина О. В., Серебряков А. А. Автоматизированное проектирование технологических процессов с использованием ЭВМ // Горный информационно-аналитический бюллетень (научно-технический журнал). 2009. № 10. С. 220–222.

202. Quad-morphing: advancing front quad meshing using triangle tranformations / S. J. Owen, M. L. Staten, S. A. Canann, S. Saigal. URL: <http://www.andrew.cmu.edu/user/sowen/qm/rt98quad.htm>.

203. Owen S. J., Staten M. L., Canann S. A. Q-Morph: an indirect approach to advancing front quad meshing // International Journal for Numerical Methods in Engineering. 1999. Vol. 44, Iss. 9. P. 1317–1340. DOI: 10.1002/(SICI)1097-0207(19990330)44:9<1317::AID-NME532>3.0.CO;2-N

204. Owen S. J., Saigal S. H-Morph: an indirect approach to advancing front hex meshing // International Journal for Numerical Methods in Engineering. 2000. Vol. 49, Iss. 1–2. P. 289–312. DOI: 10.1002/1097-0207(20000910/20)49:1/2<289::AID-NME934>3.0.CO;2-L

205. Owen S. J. Hex-domain mesh generation using 3D constrained triangulation // Computer Aided Design. 2001. Vol. 33, Iss. 3. P. 211–220.

206. Forming and resolving wedges in the spatial twist continuum / T. D. Blacker, S. A. Mitchell, T. J. Tautges [and others] // Engineering with Computers. 1997. Vol. 13, Iss. 1. P. 35–47.

207. Murdoch P., Benzley S. E. The spatial twist continuum // International Meshing Roundtable: Proceedings of The 4th International Conference (Albuquerque,

New Mexico, U.S.A., 16–17 October 1995). Sandia: Sandia National Laboratories, 1995. P. 243–251.

208. The spatial twist continuum: A connectivity based method for representing all-hexahedral finite element meshes / P. Murdoch, S. E. Benzley, T. D. Blacker, S. A. Mitchell // *Finite Elements in Analysis and Design*. 1997. Vol. 28, Iss. 2. P. 137–149.

209. Folwell N. T., Mitchell S. A. Reliable Whisker Weaving via Curve Contraction // *International Meshing Roundtable: Proceedings of the 7th International Conference*, Dearborn (Michigan, U.S.A., 26–28 October 1998). Sandia: Sandia National Laboratories, 1998. P. 365–378.

210. Tautges T. J., Mitchell S. A. Progress report on the whisker weaving all-hexahedral meshing algorithm // *Numerical Grid Generation in Computational Field Simulations: Proceedings of the 5th International Conference* (Mississippi State University, 1–5 April 1996). Mississippi: Mississippi State University, 1996. P. 659–670.

211. Tautges T. J., Blacker T. D., Mitchell S. A. The whisker weaving algorithm: a connectivity based method for constructing all-hexahedral finite element meshes // *International Journal for Numerical Methods in Engineering*. 1996. Vol. 39, Iss. 19. P. 3327–3350. DOI: 10.1002/(SICI)1097-0207(19961015)39:19<3327::AID-NME2>3.0.CO;2-H

212. Schneiders R. A grid-based algorithm for the generation of hexahedral element meshes // *Engineering with Computers*. 1996. Vol. 12, Iss. 3. P. 168–177.

213. Schneiders R., Schindler R., Weiler F. Octree-based generation of hexahedral element meshes // *International Meshing Roundtable: Proceedings of the 5th International Conference* (Pittsburgh, Pennsylvania, U.S.A., 10–11 October 1996). Sandia: Sandia National Laboratories, 1996. P. 205–216.

214. Schneiders R. Octree-based hexahedral mesh generation // *International Journal of Computational Geometry & Applications*. 2000. Vol. 10, Iss. 4. P. 383–398.

215. Ives D. Geometric grid generation. NASA CP-3291. Cleveland, 1995. P. 535–546.

216. Taghavi R. Automatic, parallel and fault tolerant mesh generation from CAD // *Engineering with Computers*. 1996. Vol. 12, Iss. 3. P. 178–185.

217. Ito Y., Shih A. M., Soni B. K. Octree-based reasonable-quality hexahedral mesh generation using a new set of refinement templates // *International Journal for Numerical Methods in Engineering*. 2009. Vol. 77, Iss. 13. P. 1809–1833. DOI: 10.1002/nme.2470

218. Liang X., Ebeida M. S., Zhang Y. Guaranteed-quality all-quadrilateral mesh generation with feature preservation // *Computer Methods in Applied Mechanics and Engineering*. 2010. Vol. 199. P. 2072–2083.

219. Marechal L. Advances in octree-based all-hexahedral mesh generation: handling sharp features // *International Meshing Roundtable: Proceedings of the 18th International Conference* (Salt Lake City, Utah, USA, 25–28 October 2009). Springer-Verlag, 2009. P. 64–84.

220. Zienkiewicz O. C., Taylor R. L., Fox D. D. The finite element method for solid and structural mechanics, Seventh Edition. London: Butterworth-Heinemann, 2013. 672 P.

221. Zheng Y., Lewis R. W., Gethin D. T. Three-dimensional unstructured mesh generation: Part 1. Fundamental aspects of triangulation and point creation // *Computer*

Methods in Applied Mechanics and Engineering. 1996. Vol. 134. P. 249–268.

222. Zheng Y., Lewis R. W., Gethin D. T. Three-dimensional unstructured mesh generation: Part 2. Surface Meshes // Computer Methods in Applied Mechanics and Engineering. 1996. Vol. 134. P. 269–284.

223. Lau T. S., Lo S. H. Finite element mesh generation over analytical curved surfaces // Computers & Structures. 1996. Vol. 59, Iss. 2. P. 301–309. DOI: 10.1016/0045-7949(95)00261-8

224. Miranda A. C. O., Martha L. F. Mesh generation on high-curvature surfaces based on a background quadtree structure // International Meshing Roundtable: Proceedings of the 11th International Conference (Ithaca, New York, 15–18 September 2002). Ithaca: Springer-Verlag, 2002. P. 333–342.

225. Miranda A. C. O., Lira W. W. M., Cavalcante-Neto J. B. A three-dimensional adaptive mesh generation approach using geometric modeling with multi-regions and parametric surfaces // Journal of Computing and Information Science in Engineering. 2013. Vol. 13, Iss. 2. P. 021002-1–021002-13. DOI: 10.1115/1.4024106

226. Miranda A. C. O., Lira W. W. M., Marques R. C. Finite element mesh generation for subsurface simulation models // Engineering with Computers. 2015. Vol. 31, Iss. 2. P. 305–324. DOI: 10.1007/s00366-014-0352-3

227. Hartmann E. A marching method for the triangulation of surfaces // The Visual Computer. 1998. Vol. 14. P. 95–108.

228. Chew L. P. Guaranteed-quality mesh generation for curved surfaces // The 9th Annual Symposium on Computational Geometry: Proceedings of the International Conference (New York, NY, USA, 18–21 May 1993). New York: ACM, 1993. P. 274–280.

229. Chen H., Bishop J. Delaunay triangulation for curved surfaces // International Meshing Roundtable: Proceedings of the 6th International Conference (Park City, UT, USA, 13–15 October 1997). Sandia: Sandia National Laboratories, 1997. P. 115–127.

230. Gabriel-constrained parametric surface triangulation / O. E. Ruiz, C. Cadavid, J. G. Lalinde [and others] // International Journal of Mathematical, Computational, Physical, Electrical and Computer Engineering. 2008. Vol. 2, No 8. P. 568–575. DOI: 10.5772/7401

231. Lorensen W. E., Cline H. E. Marching cubes: a high resolution 3D surface construction algorithm // Computer Graphics. 1987. Vol. 21, Iss. 4. P. 163–169.

232. Akkouché S., Galin E. Adaptive implicit surface polygonization using marching triangles // Computer Graphics Forum. 2001. Vol. 20, Iss. 2. P. 67–80. DOI: 10.1111/1467-8659.00479

233. Crespín B., Guitton P., Schlick C. Efficient and accurate tessellation of implicit sweep objects // Constructive Solid Geometry. 1998. P. 49–63.

234. Chinnaswamy C., Amadei B., Illangasekare T. H. A new method for finite element transitional mesh generation // International Journal for Numerical Methods in Engineering. 1991. Vol. 31, Iss. 7. P. 1253–1270. DOI: 10.1002/nme.1620310703

235. Composite mappings for planar mesh generation / R. H. Crawford, W. N. Waggenspack Jr., D. C. Anderson, M. J. Bailey // International Journal for Numerical Methods in Engineering. 1987. Vol. 24, Iss. 12. P. 2241–2252.

DOI: 10.1002/nme.1620241202

236. Wellford L. C., Gorman M. R. A finite element transitional mesh generation procedure using sweeping functions // *International Journal for Numerical Methods in Engineering*. 1988. Vol. 26, Iss. 12. P. 2623–2643. DOI: 10.1002/nme.1620261204

237. Zienkiewicz O. C., Phillips D. V. An automatic mesh generation scheme for plane and curved surfaces by ‘isoparametric’ co-ordinates // *International Journal for Numerical Methods in Engineering*. 1971. Vol. 3, Iss. 4. P. 519–528. DOI: 10.1002/nme.1620030407

238. Liu Y. C., Elmaraghy H. A., Zhang K. An expert system for forming quadrilateral finite elements // *Engineering Computations*. 1990. Vol. 7, Iss. 3. P. 249–257. DOI: 10.1108/eb023812

239. Talbert J. A., Parkinson A. R. Development of an automatic two-dimensional finite element mesh generator using quadrilateral elements and Bezier curve boundary definition // *International Journal for Numerical Methods in Engineering*. 1990. Vol. 29, Iss. 7. P. 1551–1567. DOI: 10.1002/nme.1620290712

240. Lo S. H. Generating quadrilateral elements on plane and over curved surfaces // *Computers and Structures*. 1989. Vol. 31. P. 421–426.

241. Lee C. K., Lo S. H. A new scheme for the generation of a graded quadrilateral mesh // *Computers & Structures*. 1994. Vol. 52, Iss. 5. P. 847–857. DOI: 10.1016/0045-7949(94)90070-1

242. Johnston B. P., Sullivan J. M., Kwasnik A. Automatic conversion of triangular finite element meshes to quadrilateral elements // *International Journal for Numerical Methods in Engineering*. 1991. Vol. 31, Iss. 1. P. 67–84. DOI: 10.1002/nme.1620310105

243. Lau T. S., Lo S. H., Lee C. K. Generation of quadrilateral mesh over analytical curved surfaces // *Finite Elements in Analysis and Design*. 1997. Vol. 27, Iss. 3. P. 251–272. DOI: 10.1016/S0168-874X(97)00015-2

244. Field D. A. Laplacian smoothing and Delaunay triangulation // *Communications in Applied Numerical Methods*. 1988. Vol. 4. P. 709–712.

245. Zhou T., Shimada K. An angle-based approach to two-dimensional mesh smoothing // *International Meshing Roundtable: Proceedings of the 9th International Conference (New Orleans, U.S.A., 2–5 October 2000)*. 2000. Sandia: Sandia National Laboratories. P. 373–384.

246. Chen L. Mesh smoothing schemes based on optimal Delaunay triangulations // *International Meshing Roundtable: Proceedings of the 13th International Conference (Williamburg, U.S.A., 19–22 September 2004)*. Sandia: Sandia National Laboratories, 2004. P. 109–120.

247. Hansbo P. Generalized Laplacian smoothing of unstructured grids // *Communications in Numerical Methods in Engineering*. 1995. Vol. 11. P. 455–464.

248. Canann S. A., Tristano J. R., Staten M. L. An approach to combined Laplacian and optimization-based smoothing for triangular, quadrilateral, and quadrilateral dominant meshes // *International Meshing Roundtable: Proceedings of the 7th International Conference (Dearborn, Michigan, U.S.A., 26–28 October 1998)*. Sandia: Sandia National Laboratories, 1998. P. 479–494.

249. Freitag L. A. On combining Laplacian and optimization-based mesh smoothing techniques // AMD Trends in Unstructured Mesh Generation. 1997. Vol. 220. P. 37–43.
250. Freitag L. A., Plassmann P. E. Local optimization-based untangling algorithms for quadrilateral meshes // International Meshing Roundtable: Proceedings of the 10th International Conference (Newport Beach, U.S.A., 7–10 October 2001). Sandia: Sandia National Laboratories, 2001. P. 397–406.
251. Parthasarathy V. N., Kodiyalam S. A constrained optimization approach to finite element mesh smoothing // Finite Elements in Analysis and Design. 1991. Vol. 9, Iss. 4. P. 309–320.
252. Barrera-Sánchez P., González-Flores G. F., Dominguez-Mota F. J. Robust discrete grid generation on plane irregular regions // Computational Mathematics and Mathematical Physics. 2003. Vol. 43, Iss. 6. P. 845–853.
253. Castillo J. E. An adaptive direct variational grid generation method // Computers & Mathematics with Applications. 1991. Vol. 21, Iss. 5. P. 57–64.
254. Khattri S. K. An effective quadrilateral mesh adaptation // Journal of Zhejiang University Science A. 2006. Vol. 7, Iss. 12. P. 2018–2021.
255. Khattri S. K. An adaptive quadrilateral mesh in curved domains // Serdica Journal of Computing. 2009. Vol. 3. P. 248–268.
256. Tinoco-Ruiz J.-G., Barrera-Sánchez P., Cortes-Médina A. Some properties of area functionals in numerical grid generation // International Meshing Roundtable: Proceedings of the 10th International Conference (Newport Beach, U.S.A., 7–10 October 2001). Sandia: Sandia National Laboratories, 2001. P. 43–54.
257. Amenta N., Bern M., Eppstein D. Optimal point placement for mesh smoothing // Journal of Algorithms. 1999. Vol. 30, Iss. 2. P. 302–322. DOI: 10.1006/jagm.1998.0984
258. Freitag L. A., Ollivier-Gooch C. A comparison of tetrahedral mesh improvement techniques // International Meshing Roundtable: Proceedings of the 5th International Conference (Pittsburgh, Pennsylvania, U.S.A., 10–11 October 1996). Sandia: Sandia National Labs, 1996. P. 87–100.
259. Chen Z. Combined Laplacian and optimization-based smoothing for quadratic mixed surface meshes // International Meshing Roundtable: Proceedings of the 12th International Conference (Santa Fe, U.S.A., 14–17 September 2003). Sandia: Sandia National Laboratories, 2003. P. 201–213.
260. Khattri S. K. Computationally efficient technique for nonlinear poisson-boltzmann equation // Lecture Notes in Computer Science. 2006. Vol. 3991. P. 860–883.
261. Anderson B. D., Benzley S. E., Owen S. J. Automatic all quadrilateral mesh adaption through refinement and coarsening // International Meshing Roundtable: Proceedings of The 18th International Conference (Salt Lake City, UT, USA, 25–28 October 2009). Sandia: Sandia National Laboratories, 2009. P. 557–574. DOI: 10.1007/978-3-642-04319-2_32
262. Feature sensitive surface extraction from volume data / L. P. Kobbelt, M. Botsch, U. Schwanke, H.-P. Seidel // SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques (Los Angeles, CA, USA, 12–17 August 2001). ACM Press, 2001. P. 57–66. DOI:

10.1145/383259.383265

263. Ohtake Y., Belyaev A. Dual/primal mesh optimization for polygonized implicit surfaces // Solid modeling and applications: Proceedings of The seventh ACM symposium (Saarbrücken, Germany, 17–21 June 2002). ACM Press, 2002. P. 171–178. DOI: 10.1145/566282.566308

264. Ohtake Y., Belyaev A., Pasko A. Dynamic mesh optimization for polygonized implicit surfaces with sharp features // The Visual Computer. 2003. Vol. 19, No 2-3. P. 115–126. DOI: 10.1007/s00371-002-0181-z116

265. Hoppe H., DeRose T., Duchamp T. Mesh optimization // SIGGRAPH '93: Proceedings the 20th annual conference on Computer graphics and interactive techniques (Anaheim, CA, USA, 02–06 August 1993). ACM Press, 1993. P. 19–26. DOI: 10.1145/166117.166119

266. Igarashi T., Hughes J. F. Smooth meshes for sketch-based freeform modeling // SIGGRAPH'06: The 33rd International conference on Computer graphics and interactive techniques (Boston, Massachusetts, USA, 1–3 August 2006). 2006. Article No. 13. DOI: 10.1145/1185657.1185774

267. Wilhelms J., van Gelder A. Octrees for faster isosurface generation // ACM Transactions on Graphics. 1992. Vol. 11, Iss. 3. P. 201–22. DOI: 10.1145/130881.130882

268. Shekhar R., Fayyad E., Yagel R. Octree-based decimation of marching cubes surfaces // Visualization'96: Proceedings of the IEEE International Conference (San Francisco, CA, USA, 27 Oct.–1 Nov. 1996). IEEE, 1996. P. 335–342. DOI: 10.1109/VISUAL.1996.568127

269. Shu R., Zhou C., Kankanhalli M. S. Adaptive marching cubes // The Visual Computer. 1995. Vol. 11, Iss. 4. P. 202–217. DOI: 10.1007/BF01901516

270. Ju T., Losasso F., Schaefer S., Warren J. Dual contouring of Hermite data // SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques (San Antonio, TX, USA, 21–26 July 2002). ACM Press, 2002. P. 339–346. DOI: 10.1145/566570.566586

271. Boubekur T., Schlick C. A flexible kernel for adaptive mesh refinement on GPU // Computer Graphics Forum. 2008. Vol. 27, Iss. 1. P. 102–114. DOI: 10.1111/j.1467-8659.2007.01040.x

272. Hilton A., Illingworth J. Marching triangles: Delaunay implicit surface triangulation. Technical Report EPSRC-GR/K04569. University of Surrey, Guildford, UK, 1997. 12 P.

273. Čermák M., Skala V. Polygonisation of disjoint implicit surfaces by the adaptive edge spinning algorithm // International Journal of Computational Science and Engineering. 2007. Vol. 3, No 1. P. 45–52. DOI: 10.1504/IJCSE.2007.014464

274. Dey T. K., Levine J. A. Delaunay meshing of isosurfaces // The Visual Computer. 2008. Vol. 24, Iss. 6. P. 411–422. DOI: 10.1007/s00371-008-0224-1

275. Fayolle P. A., Pasko A. Optimized surface discretization of functionally defined multi-material objects // Advances in Engineering Software. 2012. Vol. 45, Iss. 1. P. 301–312. DOI: 10.1016/j.advengsoft.2011.10.007

276. Huang H., Ascher U. Surface mesh smoothing, regularization, and feature detection // SIAM Journal on Scientific Computing. 2008. Vol. 31, Iss. 1. P. 74–93.

DOI: 10.1137/060676684

277. Multi-view stereo with single-view semantic mesh refinement / A. Romanoni, M. Ciccone, F. Visin, M. Matteucci // IEEE International Conference Computer Vision Workshop: Proceedings of the International Conference, Venice, Italy, 22–29 October 2017. 2017. P. 706–715. DOI: 10.1109/ICCVW.2017.89

278. Кравченко В. Ф., Рвачев В. Л. Алгебра логики, атомарные функции и вейвлеты в физических приложениях. Москва: ФИЗМАТЛИТ, 2006. 416 с.

279. Ricci A. A constructive geometry for computer graphics // The computer journal. 1973. Vol. 16, No 2. P. 157–163.

280. Organick E. I. A Fortran IV Primer. Addison-Wesley, 1967. 263 P.

281. Schneider Ph., Eberly D. H. Geometric Tools for Computer Graphics (The Morgan Kaufmann Series in Computer Graphics). San Francisco: Morgan Kaufmann Publishers, Elsevier Science, 2003. 1056 p.

282. Shimrat M. Algorithm 112: Position of Point Relative to Polygon // Communications of the ACM. 1962. Vol. 5, Iss. 8. P. 434. DOI: 10.1145/368637.368653

283. Fazil J., Jayakumar V. Investigation of Airfoil Profile Design using Reverse Engineering Bezier Curve // ARPN Journal of Engineering and Applied Sciences. 2011. Vol. 6, No 7. P. 43–52.

284. Parasaram R. K. N., Charyulu T. N. Airfoil Profile Design by Reverse Engineering Bezier Curve // International Journal of Mechanical Engineering and Robotics Research. 2012. Vol. 1, No 3. P. 410–420.

285. Salunke N. P., Ahamad R. A., Channiwala S. A. Airfoil Parameterization Techniques: A Review // American Journal of Mechanical Engineering. 2014. Vol. 2, No 4. P. 99–102. DOI: 10.12691/ajme-2-4-1

286. Голованов Н. Н. Геометрическое моделирование. Москва: Academia, 2011. 272 с.

287. Baumgart B. G. A polyhedron representation for computer vision // AFIPS'75: Proceedings of National Computer Conference and Exposition (New York, USA, 1975). ACM, 1975. P. 589–596.

288. Muuss M. J., Butler L. A. Combinatorial solid geometry, boundary representations, and non-manifold geometry // State of the Art in Computer Graphics: Visualization and Modeling. 1991. Vol. 1. P. 185–223.

289. Campagna S., Kobbelt L., Seidel H.-P. Directed Edges: A Scalable Representation for Triangle Meshes // Journal of Graphics Tools. 1998. Vol. 3, Iss. 4. P. 1–11.

290. Kettner L. Using generic programming for designing a data structure for polyhedral surfaces // Computational Geometry: Theory and Applications. 1999. Vol. 13, Iss. 1. P. 65–90.

291. Alumbaugh T. J., Jiao X. Compact array-based mesh data structure // International Meshing Roundtable: Proceedings of The 14th International Conference (San Diego, California, USA, 11–14 September 2005). Sandia National Laboratories, Springer-Verlag, 2005. P. 585–504.

292. Circular incident edge list: a data structure for rendering complex unstructured grids / B. Levy, G. Caumon, S. Conreux, X. Cavin // Conference on

Visualization: Proceedings of the Conference on Visualization (Washington, DC, USA, 2001). IEEE Computer Society, 2001. P. 191–198.

293. Sieger D., Botsch M. Design, implementation, and evaluation of the surface_mesh data structure // International Meshing Roundtable: Proceedings of the 20th International Conference (Paris, France, 23–26 October 2011). Sandia: Sandia National Laboratories, 2011. P. 533–550.

294. Chf: A scalable topological data structure for tetrahedral meshes / M. Lage, T. Lewiner, H. Lopes, L. Vehlo // Computer Graphics and Image Processing: Proceedings of the XIII Brazilian Symposium (Washington, DC, USA, 2005). IEEE Computer Society, 2005. P. 349–356.

295. Isenburg M., Alliez P. Compressing Hexahedral Volume Meshes // Graphical Models. 2003. Vol. 65, Iss. 4. P. 239–257.

296. Maple C. Geometric Design and Space Planning using the Marching Squares and Marching Cube Algorithms // Geometric Modeling and Graphics: Proceedings of the International Conference (London, UK, England, 16–18 July, 2003). IEEE, 2003. P. 90–95. DOI: 10.1109/GMAG.2003.1219671

297. Nielson G. M., Hamman B. The Asymptotic Decider: Resolving the Ambiguity in Marching Cubes // Visualization: Proceedings of the IEEE Conference (San Diego, California, USA, 22–25 October 1991). IEEE Computer Society Press, 1991. P. 83–91. DOI: 10.1109/VISUAL.1991.175782

298. Chernyaev E. V. Marching Cubes 33: Construction of Topologically Correct Isosurfaces. Technical Report CERN CN 95-17. CERN, 1995. 8 p.

299. Efficient Implementation of Marching Cubes' Cases with Topological Guarantees / T. Lewiner, H. Lopes, A. W. Vieira, G. Tavares // Journal of Graphics Tools. 2003. Vol. 8, Iss. 2. P. 1–15. DOI: 10.1080/10867651.2003.10487582

300. Practical Considerations on Marching Cubes 33 Topological Correctness / L. Custodio, T. Etienne, S. Pesco, C. Silva. // Computers & Graphics. 2013. Vol. 37, Iss. 7. P. 840–850. DOI: 10.1016/j.cag.2013.04.004

301. Alexandrescu A. Modern C++ Design: Generic Programming and Design Patterns Applied. USA: Addison-Wesley, 2001. 352 p.

302. Blilie C. Patterns in scientific software: an introduction // Computing in Science & Engineering. 2002. Vol. 4, Iss. 3. P. 48–53. DOI: 10.1109/5992.998640

303. Design Patterns: Elements of Reusable Object-Oriented Software / E. Gamma, R. Helm, R. Johnson, J. Vlissides. USA: Addison-Wesley, 1994. 395 p.

304. Shalloway A., Trott J. R. Design Patterns Explained: A New Perspective on Object Oriented Design. USA: Addison-Wesley, 2004. 480 p.

305. Pedoe D. Circles: A Mathematical View. The Mathematical Association of America, 1997. 138 p.

306. Möller T., Trumbore B. Fast, minimum storage ray-triangle intersection // Journal of Graphics Tools. 1997. Vol. 2, Iss. 1. P. 21–28. DOI: 10.1080/10867651.1997.10487468

307. Goerigk N., Si H. On Indecomposable Polyhedra and the Number of Steiner Points // Procedia Engineering. 2015. Vol. 124. P. 343–355. DOI: 10.1016/j.proeng.2015.10.144

308. Ruppert J., Seidel R. On the difficulty of triangulating three-dimensional

- nonconvex polyhedra // Discrete & Computational Geometry. 1992. Vol. 7, Iss. 3. P. 227–253.
309. Shewchuk J. R. Tetrahedral mesh generation by Delaunay refinement // Computational Geometry: Proceedings of The 14th Annual ACM Symposium (Minneapolis, Minnesota, USA, 7–10 June 1998). Minneapolis: ACM, 1998. P. 86–95.
310. How to Subdivide Pyramids, Prisms and Hexahedra into Tetrahedra / J. Dompierre, P. Labbé, M.-G. Vallet, R. Camarero // International Meshing Roundtable: Proceedings of the 8th International Conference (Lake Tahoe, California, USA, 10–13 October 1999). Sandia: Sandia National Laboratories, 1999. P. 195–204.
311. Тимошенко С.П. Гудьер Дж. Теория упругости. Москва: Наука, 1975. 576 с.
312. Reddy J. N. Mechanics of Laminated Composite Plates and Shells. Theory and Analysis. Boca Raton, London, New York, Washington, D.C. : CRC Press, 2003. 854 p.
313. Reissner E. The effect of transverse shear deformation on the bending of elastic plates. ASME Journal of Applied Mechanics. 1945. Vol. 12. P. 68–77.
314. A smoothed finite element method for shell analysis / N. Nguyen-Thanh, T. Rabczuk, H. Nguyen-Xuan, S. P. A. Bordas // Comput. Methods Appl. Mech. Engrg. 2008. Vol. 198. P. 165–177. DOI: 10.1016/j.cma.2008.05.029
315. Закиров Г. В. Некоторые задачи устойчивости трехслойных конических оболочек // Исслед. по теор. пластин и оболочек. 1967. Вып. 5. С. 257–270.
316. Амбарцумян С. А. Общая теория анизотропных оболочек. Москва: Наука, 1974. 448 с.
317. Григолюк Э. И., Чулков П. П. Устойчивость и колебания трехслойных оболочек. Москва: Машиностроение, 1973. 172 с.
318. Григоренко Я. М. Изотропные и анизотропные слоистые оболочки вращения переменной жесткости. К.: Наук.думка, 1973. 228 с.
319. Андреев А. Н., Немировский Ю. В. Многослойные анизотропные оболочки и пластины: Изгиб, устойчивость, колебание. Новосибирск: Наука, 2001. 288 с.
320. Сухинин С. Н. Прикладные задачи устойчивости многослойных композитных оболочек. Москва: Физматлит, 2010. 248 с.
321. Tertel E., Kuryło P. The stability of the sandwich conical shell panel – the stress state analysis // Tehnički vjesnik. 2017. Vol. 24, No. 1. P. 55–60.
322. The analytical solution of the buckling of composite truncated conical shells under combined external pressure and axial compression / M. A. Boorboor Ajdari, S. Jalili, M. Jafari [та інші] // Journal of Mechanical Science and Technology. 2012. No. 26(9). P. 2783–2791.
323. Вольмир А. С. Устойчивость деформируемых систем. Москва: Наука, 1967. 984 с.
324. Преображенский И. Н., Грищак В. З. Устойчивость и колебания конических оболочек. Москва: Машиностроение, 1986. 240 с.
325. Фрёман Н., Фрёман П. У. ВКБ-приближение. Москва: Мир, 1967. 166 с.
326. Geer J. F., Andersen C. M. A hybrid perturbation – Galerkin technique with application to slender body theory // SIAM J. Appl. Mech. 1989. Vol. 49. P. 344–361.
327. Грищак В. З. Гібридні асимптотичні методи та техніка їх застосування. Запоріжжя: ЗНУ, 2009. 225 с.
328. Грищак В. З., Дьяченко Н. Н. Определение областей устойчивости конической оболочки при комбинированном нагружении на базе гибридного

асимптотического подхода // Вісник Запорізького національного університету. Фізико-математичні науки. 2017. № 2. С. 33–46.

329. Gristchak V. Z., Dimitrijeva E. M. A Hybrid WRB-Galerkin Method and its Using to Applied Mechanics Problems // The scientific journal FACTA UNIVERSITATIS. Series: Mechanics, Automatic Control and Robotics. 1998. Vol. 2, No. 8. P. 709–713.

330. Алфутов Н. А., Зиновьев Н. А., Попов Б. Г. Расчет многослойных пластин и оболочек из композиционных материалов. М.: Машиностроение, 1984. 264 с.

331. Пановко Я. Г., Губанова И. И. Устойчивость и колебания упругих систем. М.: Наука, 1979. 384 с.

332. Каудерер Г. Нелинейная механика. М.: Изд-во иностр. Лит., 1961. 777 с.

333. Bank R. E. PLTMG: A Software Package for Solving Elliptic Partial Differential Equations: Users' Guide 8.0. SIAM, 1998. 155 p. DOI: 10.1137/1.9780898719635

334. Fully automatic adaptive mesh refinement integrated into the solution process / J. R. Tristano, Z. Chen, D. A. Hancq, W. Kwok // International Meshing Roundtable: Proceedings of the 12th International Conference (Santa Fe, New Mexico, U.S.A., 14–17 September 2003). Sandia National Laboratories, 2003. P. 307–314.

335. Pointer J. Understanding Accuracy and Discretization Error in an FEA Model. URL: <https://www.ansys.com/-/media/Ansys/corporate/resourcelibrary/conference-paper/2004-Int-ANSYS-Conf-54.PDF> (дата звернення: 19.11.2018).

336. Cickovski Tr., Matthey Th., Izaguirre J. A. Design Patterns for Generic Object-Oriented Scientific Software. Technical Report TR 2004-29. Notre Dame: University of Notre Dame, 2004. 9 P.

337. Coplien J., Hoffman D., Weiss D. Commonality and variability in software engineering // IEEE Software. 1998. Vol. 15, Iss. 6. P. 37–45.

338. Knyazev A. V., Lashuk I. Steepest Descent and Conjugate Gradient Methods with Variable Preconditioning // SIAM Journal on Matrix Analysis and Applications. 2008. Vol. 29, Iss. 4. P. 1267–1280. DOI: 10.1137/060675290

339. Notay Y. Flexible Conjugate Gradient // SIAM Journal on Scientific Computing. 2001. Vol. 22, Iss. 4. P. 1444–1460. DOI: 10.1137/S1064827599362314

340. Норенков И. П., Маничев В. Б. Основы теории и проектирования САПР: учебник для втузов по специальности «Вычислительные машины, комплексы, системы и сети». Москва: Высшая школа, 1990. 335 с.

341. ECMAScript Language Specification. URL: <http://www.ecma-international.org/ecma-262/5.1/> (дата звернення 01.08.2019).

342. IEEE Standard for Floating-Point Arithmetic. URL: <http://ieeexplore.ieee.org/servlet/opac?punumber=4610933> (дата звернення: 24.08.2020).

343. Message broker. URL: https://en.wikipedia.org/wiki/Message_broker (дата звернення: 10.03.2018).

344. Кластер (група комп'ютерів). URL: <http://jak.iblog.in.ua/articles/klaster-grupa-komp-juteriv-se.html> (дата звернення: 08.03.2018).

345. Алгоритми: побудова й аналіз / Т. Кормен, Т. Лейзерсон, Р. Ривестр, К. Штайн. Москва :Вільямс, 2014. 1290 с.

Наукове видання

ISBN 978-966-992-217-5

ЧОПОРОВ Сергій Вікторович
КУДІН Олексій Володимирович
ПАНАСЕНКО Євген Валерійович
ГРИЩАК Дмитро Дмитрович
ІГНАТЧЕНКО Марія Сергіївна

**МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ ІНЖЕНЕРНОГО АНАЛІЗУ
ОБ'ЄКТІВ АЕРОКОСМІЧНОЇ ТЕХНІКИ НА БАЗІ
ХМАРНИХ ТЕХНОЛОГІЙ**

Монографія

За науковою редакцією С. В. Чопорова

Коректор: Є. В. Панасенко
Відповідальний за випуск: О. В. Кудін

В авторській редакції

Рекомендовано до друку рішенням
Вченої ради Запорізького національного університету,
протокол № 2 від 29.09.2020 р.

ВИДАВНИЦТВО

Видавничий дім «Гельветика»
вул. Паровозна, 46а, офіс 105, м. Херсон, Україна, 73034,
тел.: +38(0552)39-95-80,
e-mail: mailbox@helvetica.com.ua

Свідоцтво Державного комітету інформаційної політики, телебачення та радіомовлення
України про внесення суб'єкта видавничої справи до Державного реєстру видавців,
виготовлювачів і розповсюджувачів видавничої продукції ДК № 6424 від 04.10.2018 р.

Підписано до друку 30.09.2020 р. Формат 60×84 1/16. Папір офсетний.

Гарнітура Times New Roman.

Умовних друкованих аркушів: 17,44. Наклад: 300 примірників.

Ціна договірна.