

Міністерство освіти і науки України
Львівський національний університет імені Івана Франка
Факультет прикладної математики та інформатики
Кафедра прикладної математики

Дипломна робота

Реалізація абстракцій у програмному забезпеченні на основі методу скінченних елементів

Виконала: студентка групи ПМП-42
спеціальності
113-прикладна математика
Курницька Ірина Андріївна

Керівник: Ящук Юрій Олександрович

Рецензент:

Львів-2023

Зміст

Вступ	3
1 Формулювання задачі	5
2 Метод скінченних елементів	9
2.1. Матриця жорсткості	15
2.2. Розв'язування СЛАР	17
2.3. Метод Гауса для чисельного інтегрування	18
3 Програмна реалізація	23
3.1. Структура програми	23
3.2. Застосування граничних умов	26
3.3. Результати та аналіз	27
3.4. Приклад 1.	27
3.5. Приклад 2.	29
Висновок	33
Список використаної літератури	35
Додатки	36

Вступ

У сучасному світі програмне забезпечення стає невід’ємною складовою багатьох сфер діяльності. За допомогою програмних продуктів реалізуються складні завдання, спрощується робота та досягається більша продуктивність в різних галузях, починаючи від медицини та промисловості і закінчуючи науковими дослідженнями. Однак, розробка програмного забезпечення з часом стає все складнішою і вимагає від розробників нових методів інтеграції та взаємодії з існуючими програмними системами.

Одним з підходів до взаємодії з програмним забезпеченням є метод скінченних елементів. Цей метод знайшов широке застосування у численних наукових областях, зокрема в інженерії, фізиці та математиці. Він дозволяє моделювати поведінку складних систем та проводити чисельні розрахунки, використовуючи скінченну кількість елементів.

Метою даної дипломної роботи є реалізація абстракцій у програмному забезпеченні на основі методу скінченних елементів. Абстракції в програмуванні дозволяють виділити ключові концепції та функціональність програмної системи, що спрощує її розробку та підтримку. Реалізація абстракцій у програмному забезпеченні на основі методу скінченних елементів має на меті створення ефективного та зручного програмного інтерфейсу, який дозволить розробникам легко взаємодіяти з програмними системами, використовуючи метод скінченних елементів для моделювання та аналізу складних систем.

Для досягнення поставленої мети необхідно розв’язати наступні завдан-

ня:

1. Дослідити метод скінченних елементів та його застосування в програмному забезпеченні.
2. Розробити програмний інтерфейс, який забезпечить зручну взаємодію з програмним забезпеченням на основі методу скінченних елементів.
3. Реалізувати необхідні функціональні можливості для введення початкових даних, налаштування параметрів моделей та отримання результатів аналізу.
4. Провести тестування та оцінку розробленого програмного інтерфейсу з використанням реальних даних та порівняти його з існуючими аналогами.

Ця дипломна робота є актуальною, оскільки реалізація абстракцій у програмному забезпеченні на основі методу скінченних елементів сприятиме покращенню ефективності та зручності роботи з програмними системами, що використовують цей метод. Результати дослідження та розробки можуть бути використані розробниками програмного забезпечення для поліпшення своїх продуктів та підвищення їх конкурентоспроможності на ринку.

У даній роботі будуть використані наступні методи дослідження: аналіз наукової літератури, проектування програмного інтерфейсу, реалізація програмного коду, тестування та оцінка результатів.

Отже, реалізація абстракцій у програмному забезпеченні на основі методу скінченних елементів є важливим завданням, що вимагає комплексного підходу та використання сучасних розробницьких інструментів. Результати роботи сприятимуть покращенню програмного забезпечення, що використовує метод скінченних елементів, і сприятимуть подальшому розвитку цієї науково-технічної галузі.

Розділ 1

Формулювання задачі

Задача теорії пружності - одна з основних проблем механіки матеріалів, яка вивчає деформації та напруження в твердих тілах під дією зовнішнього навантаження. Основною метою задач теорії пружності є визначення поведінки твердого тіла за різних умов навантаження та встановлення допустимих меж деформації. Розв'язуючи задачі теорії пружності, можна розрахувати безпеку і довговічність конструкцій, спрогнозувати їхню поведінку за різних умов експлуатації та побудувати оптимальну конструкцію.

Задача теорії пружності має загальний вигляд рівняння руху:

$$\sigma_{ij,j} + f_i = \rho \ddot{u}_i \quad (1.1)$$

де σ_{ij} є компонентами тензора напружень, f_i - компонентами зовнішніх сил або силового поля, ρ - густиною матеріалу, а \ddot{u}_i - другою похідною вектора переміщень u_i по часу.

Отримаємо таку крайову задачу теорії пружності у загальному вигляді:

$$\begin{cases} \sigma_{ij,j} + f_i = \rho \ddot{u}_i, & x \in \Omega \\ u_i|_{\Gamma} = g_i(x), & x \in \Gamma \end{cases} \quad (1.2)$$

де u_i є компонентами вектора переміщень, Ω - область, Γ - границя.

Метод скінченних елементів є ефективним інструментом для розв'язання задач пружності. Використовуючи цей метод, ми можемо розбити нашу область на менші елементи, що дозволяє отримати більш точні значення механічних напружень у кожному вузлі.

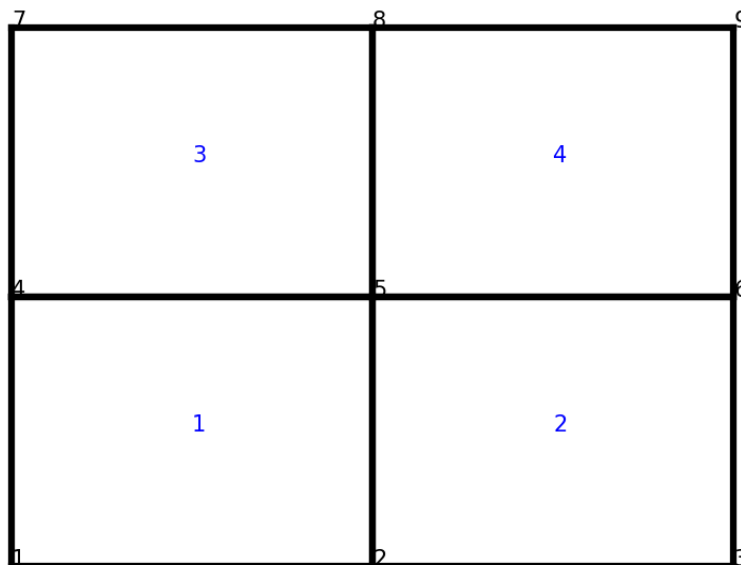


рис.1.1

У контексті задачі теорії пружності ми можемо розбити нашу область на менші елементи для отримання більш деталізованої інформації про механічні напруження. Наприклад, розбивши область на чотири менші елементи, ми отримали 9 вузлів, де ми будемо визначати значення пружності.

Для вирішення задачі пружності ми можемо використовувати метод скінченних елементів. Почнемо зі скалярного добутку функції ∇u і довільної функції v :

$$- \int_{\Omega} (\nabla u) \cdot \nabla v, d\Omega = 0. \quad (1.3)$$

Для застосування теореми Гріна до рівняння

$$- \int_{\Omega} (\nabla u) \cdot \nabla v, d\Omega = 0, \quad (1.4)$$

потрібно спочатку перетворити його у вигляд, що дозволяє використання теореми Гріна. Для цього ми можемо застосувати теорему Остроградського-Гауса до добутку $(\nabla u) \cdot \nabla v$.

Застосуємо цю теорему до виразу $(\nabla u) \cdot \nabla v$:

$$\int_{\Omega} (\nabla \cdot (\nabla u))v, d\Omega = \int_{\partial\Omega} (\nabla u) \cdot \mathbf{n}, v, ds, \quad (1.5)$$

де $\nabla \cdot (\nabla u)$ - дивергенція градієнта зміщення u , $\partial\Omega$ - границя області Ω , \mathbf{n} - вектор зовнішньої нормалі до границі, а ds - елемент довжини границі.

Отже, перетворивши вираз, ми отримали:

$$\int_{\Omega} (\nabla \cdot (\nabla u))v, d\Omega = \int_{\partial\Omega} (\nabla u) \cdot \mathbf{n}, v, ds = 0. \quad (1.6)$$

Застосовуючи тепер теорему Гріна до цього виразу, отримуємо:

$$\int_{\Omega} (\nabla \cdot (\nabla u))v, d\Omega = \int_{\Omega} \nabla u \cdot \nabla v, d\Omega - \int_{\partial\Omega} (\nabla u) \cdot \mathbf{n}, v, ds = 0. \quad (1.7)$$

Зазначимо, що виникає також умова нульового граничного умови на границі $\partial\Omega$, що дозволяє знехтувати другим доданком. Таким чином, остаточне рівняння, яке можна застосувати до теореми Гріна, матиме вигляд:

$$\int_{\Omega} \nabla u \cdot \nabla v, d\Omega = 0. \quad (1.8)$$

Отже, ми отримали вираз, що можна застосувати до теореми Гріна для розв'язання задачі пружності методом скінченних елементів.

1.0.1 (Теорема Гріна) *Нехай $u(x)$ та $v(x)$ є функціями, які мають неперервні часткові похідні на Ω та її замкненій області. Тоді має місце рівність:*

$$\int_{\Omega} (\Delta u)v d\Omega = \int_{\Omega} (\nabla u \cdot \nabla v) d\Omega + \int_{\Gamma} (u \frac{\partial v}{\partial n} - v \frac{\partial u}{\partial n}) d\Gamma, \quad (1.9)$$

де Δu - лапласіан функції u , ∇u - градієнт функції u , $\frac{\partial u}{\partial n}$ та $\frac{\partial v}{\partial n}$ - частинні похідні функцій u та v вздовж зовнішньої нормалі до границі Γ , а $d\Omega$ та $d\Gamma$ позначають елементи площі та довжини, відповідно.

1.0.2 (Теорема Остроградського-Гауса) Нехай V - обмежена область в просторі з границею ∂V і розглядається достатньо гладке векторне поле $\mathbf{F} = (F_1, F_2, F_3)$, де F_i - компоненти вектора \mathbf{F} , які мають неперервні частинні похідні другого порядку в V . Тоді теорема Остроградського-Гауса стверджує наступне:

$$\iiint_V \nabla \cdot \mathbf{F}, dV = \iint_{\partial V} \mathbf{F} \cdot \mathbf{n}, dS, \quad (1.10)$$

де $\nabla \cdot \mathbf{F}$ - дивергенція векторного поля \mathbf{F} , ∂V - замкнена поверхня, що обмежує область V , \mathbf{n} - вектор зовнішньої нормалі до поверхні, а dS - елемент площі поверхні.

Застосування теореми Гріна дозволяє нам встановити зв'язок між криволінійним інтегралом по замкнутому контуру і подвійним інтегралом по області, що обмежує цей контур. У випадку задач теорії пружності теорема Гріна використовується для апроксимації функції на скінченних елементах і отримання рівнянь, які можна обчислити у вузлах сітки для розв'язання задачі. Застосувавши її в нашому випадку, отримаємо:

$$-\int_{\Omega} \Delta u \cdot v d\Omega = \int_{\Omega} \nabla u \cdot \nabla v d\Omega - \int_{\Gamma} u \cdot \Delta v d\Gamma. \quad (1.11)$$

Якщо припустити, що функція $g(x)$ є сталою (наприклад, $g(x) = 0$), то очевидно, що другий інтеграл $\int_{\Gamma} u \cdot \Delta v d\Gamma = 0$.

Розділ 2

Метод скінченних елементів

Для розв'язування задачі методом скінченних елементів область, на якій розглядаємо задачу, потрібно наблизити до стандартного прямокутного вигляду. Отриманий прямокутник називають загальним двовимірним чотирикутним елементом, який можна легко співставляти до заданої області. На рисунку 2.1 показано батьківський елемент і його натуральні (r, s) координати та чотирикутний елемент у глобальній декартовій системі координат.

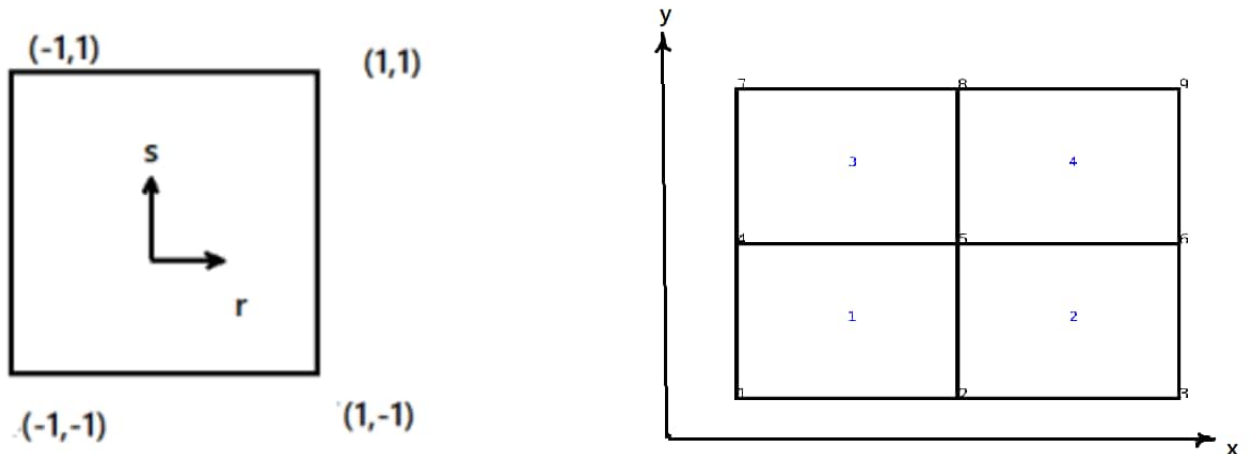


рис.2.1

Функції інтерполяції для початкового елемента можуть бути використані для геометричних функцій, якщо ми зіставимо координати так:

Внаслідок цих перетворень отримуємо нові вирази для x та y .

$$x = \sum_{i=1}^{n=4} N_i(r, s)x_i \tag{2.1}$$

$$y = \sum_{i=1}^{n=4} N_i(r, s)y_i \tag{2.2}$$

Очевидно, що ми також можемо виразити варіацію змінної поля в чотирикутнику як

$$\phi(x, y) = \phi(r, s) = \sum_{i=1}^{n=4} N_i(r, s)\phi_i \tag{2.3}$$

Оскільки інтерполяційні функції виражені в (r, s) координатах, то формально похідні, необхідні для пошуку елементів матриці жорсткості, ми можемо записати наступним чином:

$$\frac{\partial N_i}{\partial x} = \frac{\partial N_i}{\partial r} \frac{\partial r}{\partial x} + \frac{\partial N_i}{\partial s} \frac{\partial s}{\partial x} \tag{2.4}$$

$$\frac{\partial N_i}{\partial y} = \frac{\partial N_i}{\partial r} \frac{\partial r}{\partial y} + \frac{\partial N_i}{\partial s} \frac{\partial s}{\partial y} \tag{2.5}$$

З 2.3 випливає, що для обчислення часткових похідних використовуватимемо наступні формули:

$$\frac{\partial N_i}{\partial r} = \frac{\partial N_i}{\partial x} \frac{\partial x}{\partial r} + \frac{\partial N_i}{\partial y} \frac{\partial y}{\partial r} \tag{2.6}$$

$$\frac{\partial N_i}{\partial s} = \frac{\partial N_i}{\partial x} \frac{\partial x}{\partial s} + \frac{\partial N_i}{\partial y} \frac{\partial y}{\partial s} \tag{2.7}$$

, $i = 1, 4$.

Запишемо рівняння 2.6, 2.7 у матричному вигляді.

$$\begin{Bmatrix} \frac{\partial N_i}{\partial r} \\ \frac{\partial N_i}{\partial s} \end{Bmatrix} = \begin{bmatrix} \frac{\partial x}{\partial r} & \frac{\partial y}{\partial r} \\ \frac{\partial x}{\partial s} & \frac{\partial y}{\partial s} \end{bmatrix} \begin{Bmatrix} \frac{\partial N_i}{\partial x} \\ \frac{\partial N_i}{\partial y} \end{Bmatrix} \quad (2.8)$$

Ми бачимо, що вектор 2×1 з лівого боку відомий, оскільки інтерполяційні функції виражаються явно в натуральних координатах. Так само, члени в матриці коефіцієнтів 2×2 у правій частині відомі через Рівняння 2.1, 2.2. Остання, відома як матриця Якобі, позначена $[J]$.

$$[J] = \begin{bmatrix} \frac{\partial x}{\partial r} & \frac{\partial y}{\partial r} \\ \frac{\partial x}{\partial s} & \frac{\partial y}{\partial s} \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{n=4} \frac{\partial N_i}{\partial r} x_i & \sum_{i=1}^{n=4} \frac{\partial N_i}{\partial r} y_i \\ \sum_{i=1}^{n=4} \frac{\partial N_i}{\partial s} x_i & \sum_{i=1}^{n=4} \frac{\partial N_i}{\partial s} y_i \end{bmatrix} \quad (2.9)$$

Якщо можна визначити обернену матрицю до матриці Якобі, рівняння 2.8 може бути розв'язане для часткових похідних інтерполяційних функцій для отримання наступного запису:

$$\begin{Bmatrix} \frac{\partial N_i}{\partial x} \\ \frac{\partial N_i}{\partial y} \end{Bmatrix} = [J]^{-1} \begin{Bmatrix} \frac{\partial N_i}{\partial r} \\ \frac{\partial N_i}{\partial s} \end{Bmatrix} = \begin{bmatrix} I_{11} & I_{12} \\ I_{21} & I_{22} \end{bmatrix} \begin{Bmatrix} \frac{\partial N_i}{\partial r} \\ \frac{\partial N_i}{\partial s} \end{Bmatrix} \quad (2.10)$$

, де члени оберненої матриці Якобі, позначені для зручності I_{ij} .

Як ми знаємо, для отримання елементів матриці жорсткості нам потрібно обчислювати різні інтеграли. Наприклад, при обчисленні членів матриці

провідності для двовимірних теплообмінних елементів, інтеграли виглядають так:

$$\iint_{\Omega} \left(\frac{\partial N_i}{\partial x} \frac{\partial N_j}{\partial x} \right) d\Omega$$

(2.11)

і розв'язуватимуться за площею елемента у глобальних координатах. Для того, щоб перейти у систему координат (r, s) скористаємось наступною формулою, в якій підінтегральний вираз перетворюється за допомогою елементів оберненої матриці Якобі $[J]^{-1}$.

Таким чином інтеграл описаний вище по заданій площині Ω записуємо так:

$$\iint_{\Omega} \left(\frac{\partial N_i}{\partial x} \frac{\partial N_j}{\partial x} \right) d\Omega = \iint_{-1}^1 \left(I_{11} \frac{\partial N_i}{\partial r} + I_{12} \frac{\partial N_i}{\partial s} \right) \left(I_{11} \frac{\partial N_j}{\partial r} + I_{12} \frac{\partial N_j}{\partial s} \right) |J| dr ds$$

(2.12)

,

де $d\Omega = dx dy = |J| dr ds$, а $|J|$ - якобіан переходу

Розглянемо це на прикладі площини, в яку входять вузли 1, 2, 4, 5.

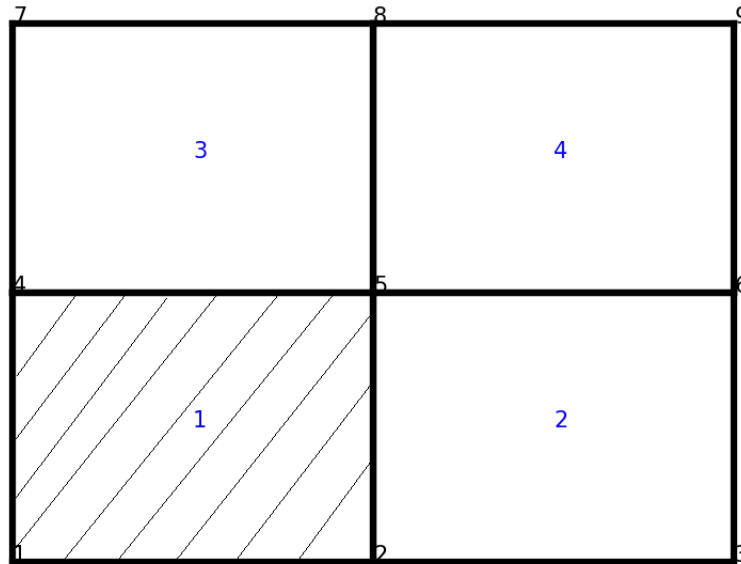


рис.2.2

Застосовуючи всі вищезгадані умови, які мають задовольняти інтерполяційна функція в кожному вузлі, отримуємо:

$$\begin{aligned}
 N_1(r, s) &= \frac{1}{4}(1 - r)(1 + s) \\
 N_2(r, s) &= \frac{1}{4}(1 + r)(1 + s) \\
 N_4(r, s) &= \frac{1}{4}(1 - r)(1 - s) \\
 N_5(r, s) &= \frac{1}{4}(1 + r)(1 - s)
 \end{aligned}
 \tag{2.13}$$

Часткові похідні x і y відносно r і s за рівняннями (2.1), (2.2) шукаємо за наступними формулами.

$$\frac{\partial x}{\partial r} = \sum_{i=1}^{n=4} \frac{\partial N_i}{\partial r} x_i = \frac{1}{4}[-(1 - s)x_4 + (1 - s)x_5 + (1 + s)x_2 - (1 + s)x_1] \tag{2.14}$$

$$\frac{\partial y}{\partial r} = \sum_{i=1}^{n=4} \frac{\partial N_i}{\partial r} y_i = \frac{1}{4} [-(1-s)y_4 + (1-s)y_5 + (1+s)y_2 - (1+s)y_1] \quad (2.15)$$

$$\frac{\partial x}{\partial s} = \sum_{i=1}^{n=4} \frac{\partial N_i}{\partial s} x_i = \frac{1}{4} [-(1-r)x_4 - (1+r)x_5 + (1+r)x_2 + (1-r)x_1] \quad (2.16)$$

$$\frac{\partial y}{\partial s} = \sum_{i=1}^{n=4} \frac{\partial N_i}{\partial s} y_i = \frac{1}{4} [-(1-r)y_4 - (1+r)y_5 + (1+r)y_2 + (1-r)y_1] \quad (2.17)$$

Тоді матриця Якобі матиме вигляд:

$$[J] = \frac{1}{4} \begin{bmatrix} (1-s)(x_5 - x_4) + (1+s)(x_2 - x_1) & (1-s)(y_5 - y_4) + (1+s)(y_2 - y_1) \\ (1-r)(x_1 - x_4) + (1+r)(x_2 - x_5) & (1-r)(y_1 - y_4) + (1+r)(y_2 - y_5) \end{bmatrix} \quad (2.18)$$

Вищезгадані значення у вузлах підставляємо в матрицю 2.18 та рахуємо визначник, який і є якобіаном переходу.

Аналогічно, можемо розглянути приклад, де область розбивається на трикутні елементи. Для цього потрібно наблизити область, на якій розглядається задача, до стандартного трикутного вигляду, який має наступний вигляд:

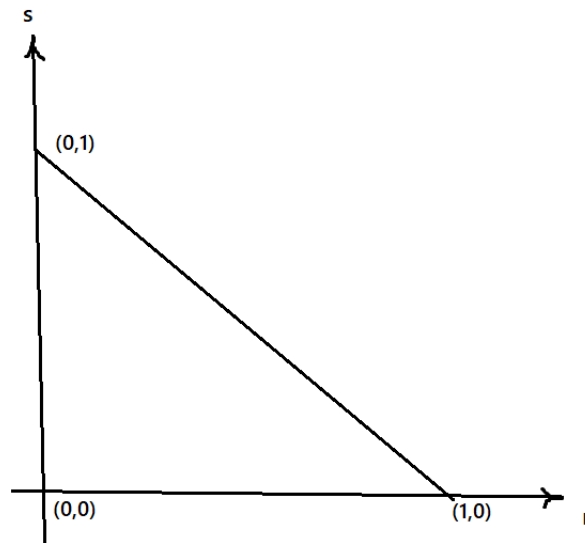


рис.2.3

2.1. Матриця жорсткості

Матриця жорсткості в методі скінченних елементів відображає взаємодію між елементами, які складають систему і визначають її поведінку. Матриця жорсткості розраховується залежно від локальної матриці жорсткості та геометрії елементів.

Відповідно до методу скінченних елементів (МСЕ), система рівнянь може бути виражена наступним чином:

$$[K]U = F$$

де $[K]$ є глобальною матрицею жорсткості, U - вектором невідомих, а F - вектором зовнішніх навантажень.

Локальна матриця жорсткості елемента використовується для побудови глобальної матриці жорсткості $[K]$. Кожен елемент системи, наприклад, трикутні елементи, чотирикутні елементи та інші типи елементів, має власну локальну матрицю жорсткості.

Локальна матриця жорсткості елемента розраховується відповідно до його рівняння, типу елемента і властивостей матеріалу. Ця матриця враховує механічні властивості матеріалу, геометрію елемента і розподіл сил всередині елемента. Локальна матриця жорсткості - це квадратна матриця, що представляє взаємодію між вузлами компонента. Прикладами локальних матриць жорсткості є коефіцієнти жорсткості, площі трикутників і чотирикутників, довжини ребер і кути нахилу ребер.

Після розрахунку локальної матриці жорсткості для кожного елемента, вони об'єднуються в глобальну матрицю жорсткості $[K]$. Це робиться шляхом розміщення локальної матриці жорсткості у відповідній позиції в глобальній матриці з урахуванням зв'язків між елементами.

Приклад локальної матриці жорсткості можна проілюструвати за допомогою простих двовимірних елементів, таких як трикутні елементи. Наприклад, припустимо, що є трикутний елемент з трьома вузлами, 1, 2 і 3. Локальна матриця жорсткості для цього елемента буде виглядати наступним чином:

$$[K_{\text{local}}] = \begin{bmatrix} k_{11} & k_{12} & k_{13} \\ k_{21} & k_{22} & k_{23} \\ k_{31} & k_{32} & k_{33} \end{bmatrix}$$

де k_{ij} - елементи локальної матриці жорсткості.

Після обчислення локальної матриці жорсткості, її елементи розміщуються у відповідних місцях глобальної матриці жорсткості $[K]$. Наприклад, якщо елемент взаємодії між вузлами 1 і 2, то елементи k_{11} , k_{12} , k_{21} і k_{22} локальної матриці жорсткості розміщуються у відповідні місця у глобальній матриці жорсткості $[K]$. Для прямокутного елемента, який складається з чотирьох вузлів, локальна матриця жорсткості матиме розмірність 8x8. У загальному випадку, локальна матриця жорсткості прямокутного елемента може мати наступний вигляд:

$$[K_{\text{local}}] = \begin{bmatrix} k_{11} & k_{12} & k_{13} & k_{14} & k_{15} & k_{16} & k_{17} & k_{18} \\ k_{21} & k_{22} & k_{23} & k_{24} & k_{25} & k_{26} & k_{27} & k_{28} \\ k_{31} & k_{32} & k_{33} & k_{34} & k_{35} & k_{36} & k_{37} & k_{38} \\ k_{41} & k_{42} & k_{43} & k_{44} & k_{45} & k_{46} & k_{47} & k_{48} \\ k_{51} & k_{52} & k_{53} & k_{54} & k_{55} & k_{56} & k_{57} & k_{58} \\ k_{61} & k_{62} & k_{63} & k_{64} & k_{65} & k_{66} & k_{67} & k_{68} \\ k_{71} & k_{72} & k_{73} & k_{74} & k_{75} & k_{76} & k_{77} & k_{78} \\ k_{81} & k_{82} & k_{83} & k_{84} & k_{85} & k_{86} & k_{87} & k_{88} \end{bmatrix}$$

Елементи локальної матриці жорсткості k_{ij} прямокутного елемента залежать від геометрії елемента, властивостей матеріалу та властивостей моделі. Зазвичай їх обчислюють шляхом інтегрування рівняння, яке враховує геометричні параметри елемента, напруження та деформації, а також параметри матеріалу, такі як модуль Юнга та коефіцієнт Пуассона.

Коли в МСЕ використовуються прямокутні елементи, локальна матриця жорсткості буде розміщена у відповідній позиції в глобальній матриці жорсткості з урахуванням зв'язків між вузлами.

Таким чином, глобальна матриця жорсткості формується шляхом збору локальних матриць жорсткості елементів та розміщення їх елементів у відповідних місцях у глобальній матриці. Цей процес визначатиме поведінку системи при заданих умовах навантаження.

2.2. Розв'язування СЛАР

В методі скінченних елементів (МСЕ), система рівнянь у вигляді лінійних алгебраїчних рівнянь (СЛАР) отримується з використанням матриці жорсткості та вектора зовнішніх навантажень.

$$[K]U = F$$

Для розв'язання СЛАР використовуються різні методи, включаючи прямі та ітераційні. Типовим прямим методом є метод декомпозиції LU, який використовує факторизацію матриці жорсткості на нижню трикутну матрицю (L) і верхню трикутну матрицю (U). Такий розклад дозволяє легко отримати розв'язок СЛАР шляхом послідовного застосування прямого та зворотного ходу.

Розв'язок СЛАР дає вектор невідомих U , який відображає значення шуканих величин (таких як переміщення і температури) у вузлах системи. Цей вектор можна використовувати для розрахунку інших параметрів та аналізу поведінки системи.

В даній роботі я використовую функцію `np.linalg.solve()` з бібліотеки NumPy для розв'язування системи лінійних алгебраїчних рівнянь. Застосування цієї функції дозволить знайти вектор U , який є розв'язком системи.

Код, що реалізує розв'язування системи, може мати наступний вигляд:

```
U = np.linalg.solve(K, F)
```

У цьому коді 'K' - матриця жорсткості після застосування відповідних граничних умов або методів скорочення, а 'F' - вектор зовнішніх навантажень. Результатом виконання коду буде вектор 'U', що містить значення невідомих змінних.

2.3. Метод Гауса для чисельного інтегрування

Знаходження значення інтеграла є важливим елементом багатьох прикладних задач. На практиці, однак, часто неможливо точно обчислити значення інтеграла, оскільки вихідна функція є складною або не може бути виражена через елементарні функції.

Одним з методів чисельного інтегрування є метод Гауса. Він полягає у знаходженні наближення для заданого інтеграла шляхом обчислення скінченних сум на заданій сітці.

Задача чисельного інтегрування полягає у знаходженні наближеного значення визначеного інтеграла

$$I = \int_a^b \rho(x)f(x) dx, \quad (2.19)$$

де $f(x)$ - задана функція, $\rho(x)$ - ваговий множник ($\rho(x) > 0$), а $[a, b]$ - інтервал інтегрування.

Наближене значення інтеграла можна отримати за допомогою скінченної суми

$$I_h = \sum_{k=0}^n c_k f(x_k), \quad (2.20)$$

де x_k - вузли, а c_k - коефіцієнти квадратурної формули. Тобто наближено згадану вище суму 2.20 можемо подати таким чином:

$$\int_a^b \rho(x)f(x) dx \approx \sum_{k=0}^n c_k f(x_k) \quad (2.21)$$

Метод Гауса підбирає вузли та коефіцієнти квадратурного рівняння так, щоб вони були точними для поліномів заданого степеня. Це дозволяє отримати наближені значення інтеграла з високим ступенем точності.

Інтерпольована квадратура є різновидом методу Гауса. Він базується на вираженні функції $f(x)$ у вигляді суми інтерполяційного полінома та залишкового члена.

Нехай маємо функцію $f(x)$ та розглядаємо інтерполяційну квадратурну формулу. Позначимо функцію $f(x)$ як суму двох частин: $\varphi(x)$ та $r(x)$, де $\varphi(x)$ - загальний інтерполяційний многочлен, а $r(x)$ - залишковий член інтерполяційної формули. Тоді можна записати

$$\int_a^b \rho(x)f(x)dx = \int_a^b \rho(x)\varphi(x)dx + \int_a^b \rho(x)r(x)dx = \sum_{k=0}^n c_k f(x_k) + R(f),$$

де $R(f)$ є формулою для залишкового члена інтерполяції, яка має вигляд:

$$|r(x)| \leq \frac{M_{n+1}}{(n+1)!} |\omega(x)|,$$

де $M_{n+1} = \max_{x \in [a,b]} |f^{(n+1)}(x)|$, $\omega(x) = (x - x_0)(x - x_1) \dots (x - x_n)$.

Таким чином, оцінка залишкового члена квадратурної формули має вигляд:

$$|R(x)| \leq \int_a^b \rho(x) f(x) \frac{M_{n+1}}{(n+1)!} |\omega(x)| dx.$$

Якщо розглянути інтерполяційний многочлен у формі Лагранжа L_n і підставити його в формулу для залишкового члена, отримаємо:

$$\begin{aligned} \int_a^b \rho(x) f(x) dx &= \int_a^b \rho(x) L_n(x) dx + R(f) \\ &= \sum_{k=0}^n f(x_k) \int_a^b \rho(x) \frac{\omega(x)}{(x - x_k) \omega'(x_k)} dx + R(f). \end{aligned}$$

Це дає нам наближену квадратурну формулу, де вагові коефіцієнти c_k можна обчислити за формулою:

$$c_k = \int_a^b \rho(x) \frac{\omega(x)}{(x - x_k) \omega'(x_k)} dx.$$

Зауваження: Якщо квадратурна формула є точною для поліномів степеня n , то $R(f) = 0$ і формула називається інтерполяційною. Зазначимо, для полінома степеня n формула (2.20) є точною, оскільки тоді $L_n(x) \equiv f(x)$. Зокрема, формула (2.20) є точною при $f(x) = x^i$, $i = \overline{0, n}$. Врахувавши цю умову, отримуємо наступну систему лінійних рівнянь:

$$\sum_{k=0}^n c_k x_k^i = \frac{b^{i+1} - a^{i+1}}{i+1}, \quad i = \overline{0, n} \quad (2.22)$$

Оскільки визначник системи (2.22) не дорівнює 0 (визначник Вандермонда), то для неї завжди єдине рішення.

Зауваження. Зрозуміло, що квадратурні формули інтерполяційного типу мають алгебраїчний степінь точності принаймні n , але виявляється, що для парних n та симетричного розташування вузлів інтегрування (відносно середини проміжку), алгебраїчний степінь точності на одиницю вищий степеня інтерполяційного полінома ($m = n + 1$).

Зауваження. Підвищення точності інтеграла зазвичай досягається розбиттям інтервалу на рівні частини. Для підвищення точності квадратурної формули може здатися природним збільшити степінь многочлена, квадратурна формула якого є правильною. Однак такий підхід не такий простий, як здається. Дійсно, розглянемо наступну квадратурну формулу.

$$S_n(f) = \frac{b-a}{2} \sum_{k=0}^n D_k f(x_k) \approx I(f) = \int_a^b f(x)p(x)dx,$$

яка є точною для будь-якого многочлена P_m степеня m :

$$I(P_m) = S(P_m).$$

Звідси

$$R(f) = I(f) - S_n(f) = R(P_m) + R(f - P_m) = R(f - P_m).$$

Має місце очевидна оцінка

$$|R(f)| \leq V_n E_m(f),$$

де

$$E_m(f) = \inf_{P_m} \sup_{[a,b]} |f(x) - P_m(x)|,$$

$$V_n = \int_a^b |p(x)|dx + \left(\frac{b-a}{2}\right) \sum_{k=0}^n |D_k|.$$

За невдалого вибору вузлів може виявитися, що для деяких (навіть аналітичних) функцій, величина V_n збільшується із зростанням n так, що це

збільшення не може компенсувати спадання E_m . Окрім того, для формул Ньютона-Котеса можна показати, що серед D_k за великих n будуть зустрічатися як додатні, так і від'ємні величини, які по модулю перевищують як завгодно велике число. Звідси випливає, що за великих n , малі похибки у значеннях функції $f(x_k)$ можуть дати велику похибку у квадратурній сумі.

Тому наведені вище формули не підходять для обчислень, де кількість вузлів (або максимальний степінь полінома, який вони правильно обчислюють) є великою. Тому часто буває вигідно розбити вихідний інтервал, застосувати квадратурну формулу до кожної частини з невеликою кількістю вузлів і додати результат до всіх частин. Іншими словами, замість звичайної формули використовується складена формула.

Розділ 3

Програмна реалізація

У цьому розділі буде описано програмну реалізацію методу скінченних елементів, яка використовується для обчислення значень матриці жорсткості і проведення пост-процесингу результатів. Метою розробки програмної реалізації є створення інструменту, що застосовує метод скінченних елементів для чисельного моделювання поведінки деформацій та напружень в механічних деталях. Цей метод дозволяє здійснювати аналіз деталей зі складною геометрією та різними матеріалами, допомагає визначити критичні зони, прогнозувати місця виникнення пошкоджень та оцінювати міцність конструкцій.

3.1. Структура програми

Мови програмування та бібліотеки: У розробці програмної реалізації використовується мова програмування Python. Для виконання обчислень та наукової візуалізації використовуються наступні бібліотеки:

- NumPy: використовується для роботи з масивами та векторами і проведення чисельних обчислень.
- Matplotlib: використовується для побудови графіків та діаграм.

- Matplotlib.colors: використовується для керування кольоровими мапами для візуалізації результатів.
- Datetime: використовується для вимірювання часу виконання програми.
- abc: використовується для реалізації абстрактних базових класів (Abstract Base Classes, ABCs) та поліморфізму типів. Абстрактні базові класи є класами, від яких не можна створювати екземпляри, але вони можуть містити абстрактні методи, які потрібно реалізувати в похідних класах.

Огляд основних кроків алгоритму: Основні кроки алгоритму програмної реалізації методу скінчених елементів включають:

1. Імпорт необхідних модулів.
2. Ініціалізація параметрів, таких як розміри, геометрія, тип елементів, значення початкових умов.
3. Створення сітки скінчених елементів.
4. Відображення вузлів та елементів на графіку.
5. Розрахунок матриць жорсткості, сили та зсуву.
6. Застосування граничних умов до матриць.
7. Редукція матриць до необхідного розміру.
8. Розв'язок системи лінійних рівнянь для знаходження переміщень.
9. Візуалізація результатів, побудова графіків напружень та переміщень.

У даній програмі демонструється використання абстрактних базових класів (ABC) у Python. ABC - це класи, які призначені для успадкування, але не для створення екземплярів безпосередньо. Вони визначають загальний інтерфейс або набір абстрактних методів, які повинні бути реалізовані конкретними підкласами. ABC надає спосіб забезпечити певну структуру або поведінку в підкласах та сприяють повторному використанню коду.

На даний момент визначено два абстрактних базових класи: `AbstractMesh` та `AbstractPostProcess`. Ці класи слугують зразками для створення конкретних підкласів, які реалізують певну функціональність.

Клас `AbstractMesh` визначає один абстрактний метод `generatemesh()`, який відповідає за генерацію сітки. Кожен клас, який успадковує `AbstractMesh`, повинен реалізувати цей метод. У коді клас `BoatMesh` є конкретним підкласом `AbstractMesh` і надає реалізацію методу `generatemesh()`. Цей клас генерує сітку на основі наданих параметрів.

Клас `AbstractPostProcess` також містить та визначає один абстрактний метод `postprocess()`, який відповідає за обробку результатів. Будь-який клас, який успадковує `AbstractPostProcess`, повинен реалізувати цей метод. Клас `BoatPostProcess` є конкретним підкласом `AbstractPostProcess` і реалізує метод `postprocess()`. Цей клас виконує післяобробку результатів, отриманих з сітки.

Клас `AbstractVisualizer` це допоміжний клас, який відповідає за створення фігури і осі для графічного відображення.

Метод `plot()` в межах цього класу приймає масиви `x`, `y` і `z` як вхідні дані та генерує графік за допомогою функцій `tripcolor()` і `plot()`.

За допомогою визначення абстрактних базових класів та конкретних підкласів забезпечується реалізація певних методів у визначений спосіб. Це сприяє організації коду, можливості розділення на компоненти та дотриманню визначеної структури.

3.2. Застосування граничних умов

Даний код виконує розрахунок методом скінченних елементів для задачі пружності. Основні функції включають в себе присвоєння граничних умов, збірку матриці жорсткості, розрахунок матриць жорсткості для кожного скінченного елемента, обчислення геометричних точок Гауса для інтегрування, обчислення градієнта функцій форми, збірку векторів зсувів та сил, оновлення значень вузлових величин з урахуванням зсувів та сил.

Граничні умови встановлюються в функції 'assign BCs' за допомогою параметру 'BC flag', який може приймати значення 'extension', 'expansion' або 'shear'. В залежності від значення 'BC flag', встановлюються відповідні умови Діріхле та/або умови Неймана на границях області.

Далі, у функції 'assemble stiffness' здійснюється збірка матриці жорсткості 'K' шляхом обчислення матриць жорсткості для кожного скінченного елемента та додавання їх відповідним частинам матриці 'K'.

У функції 'element stiffness' обчислюється матриця жорсткості для конкретного скінченного елемента. Здійснюється ітерація по гаусових точках для обчислення градієнта функцій форми, обчислення матриці Якобіану та виведення градієнтів функцій форми на основі матриці Якобіану. Застосовується матричне множення та обчислення значень відповідних елементів матриці жорсткості.

Функції 'assemble displacements', 'assemble forces' та 'update nodes' виконують збірку векторів зсувів, векторів сил та оновлення вузлових значень відповідно. Вектори зсувів та сил формуються на основі граничних умов, а значення вузлових величин оновлюються з урахуванням цих зсувів та сил.

3.3. Результати та аналіз

Базуючись на отриманих результатах будуються графіки деформацій у напрямку X (Displacement X) і напружень у напрямку X (Stress xx).

Графік Displacement X відображає переміщення вузлів моделі в напрямку X під дією прикладених навантажень. Колірна шкала на графіку відображає величину переміщення, де світлий колір відповідає великим переміщенням, а темний колір - невеликим.

Графік Stress xx відображає розподіл напружень у матеріалі в напрямку X . Колірна шкала на графіку представляє величину напружень, де світлий колір відповідає високим значенням напружень, а темний колір - низьким значенням.

Обидва графіки базуються на результатах чисельного моделювання методом скінченних елементів і дозволяють візуалізувати зміни в деформаціях та напруженнях вздовж моделі. Вони можуть бути використані для аналізу поведінки матеріалу та виявлення критичних зон, де можуть виникнути пошкодження або деформації.

Зауважимо, що результати представлені у нормалізованому вигляді, де значення деформацій та напружень були перетворені таким чином, щоб вони знаходилися в межах від 0 до 1. Це полегшує порівняння між різними точками моделі та візуальне сприйняття розподілу деформацій та напружень.

3.4. Приклад 1.

Розглянемо задачу теорії пружності, яку будемо розв'язувати методом скінченних елементів.

Задана форма нашої області є зображена на рисунку 3.1. Нам відомо, що на лівій та правій границях області діє розтягуюча сила з величиною

0,1.

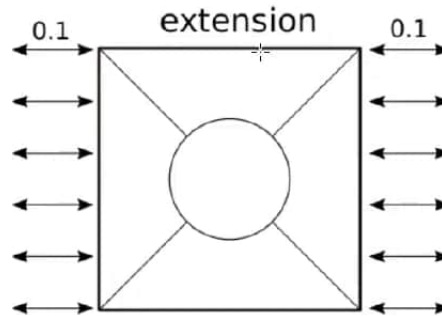


Рис. 3.1:

Завдання: Знайти деформацію та напруження в матеріалі при заданій розтягуючій силі та геометрії області.

Наша задача полягає в розв'язанні рівнянь пружності для даної геометрії з врахуванням граничних умов, які включають розтягуючі сили на границях області.

Отримаємо наступні графіки деформацій у напрямку X (Displacement X) і напружень у напрямку X (Stress xx).

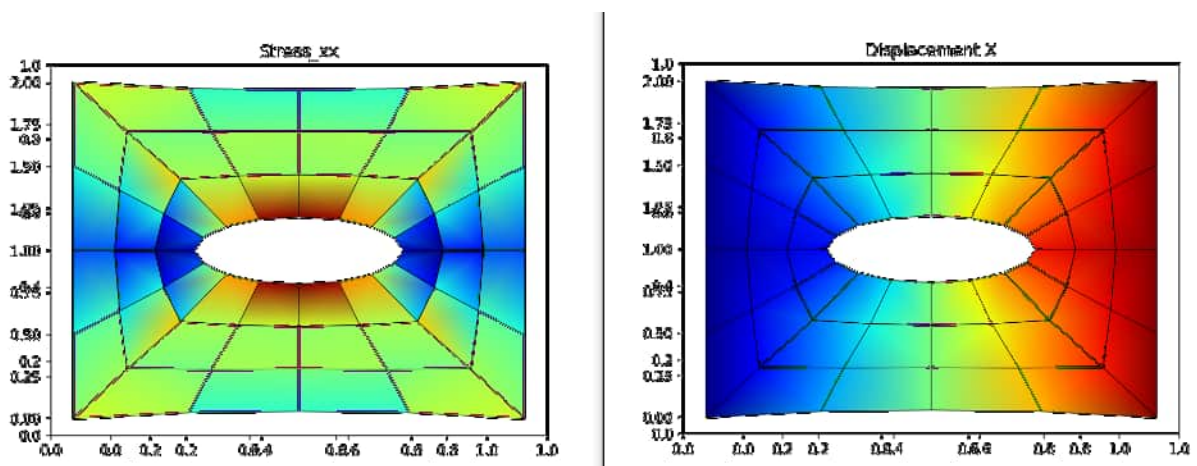


Рис. 3.2:

Спробуємо збільшити щільність сітки та змінимо елемент триангуляції

на трикутник і отримаємо такі результати:

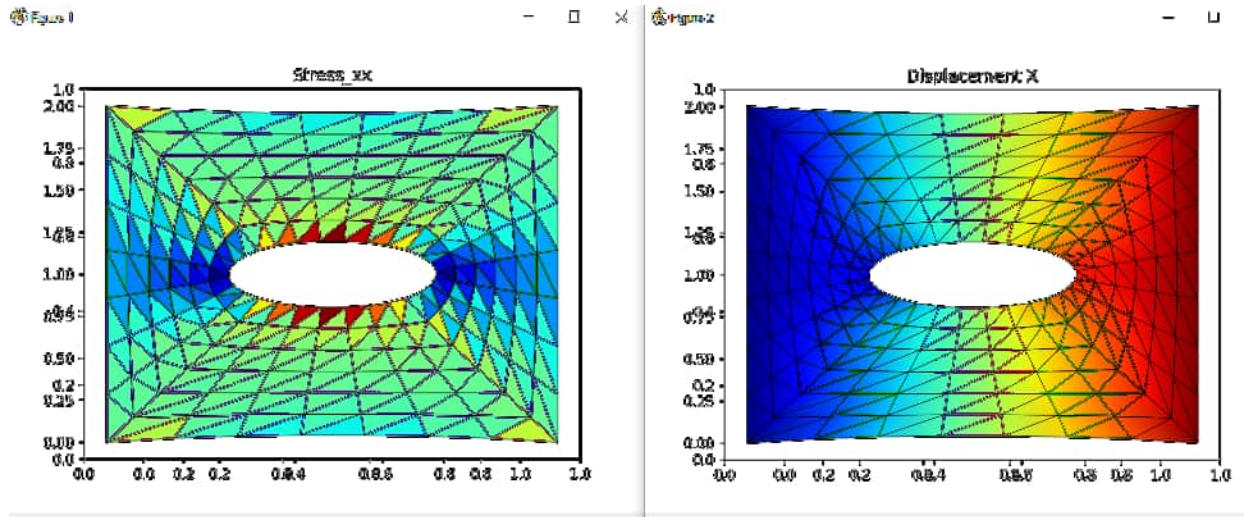


Рис. 3.3:

3.5. Приклад 2.

Розглянемо задачу теорії пружності, яку будемо розв'язувати методом скінченних елементів.

Задана форма нашої області є зображена на рисунку 3.3. Нам відомо, що на кожній з границь області діє розтягуюча сила з величиною 0,3.

Завдання: Знайти деформацію та напруження в матеріалі при заданій розтягуючій силі та геометрії області.

Наша задача полягає в розв'язанні рівнянь пружності для даної геометрії з врахуванням граничних умов, які включають розтягуючі сили на границях області.

Отримаємо такі графіки деформацій у напрямку X (Displacement X) і напружень у напрямку X (Stress xx).

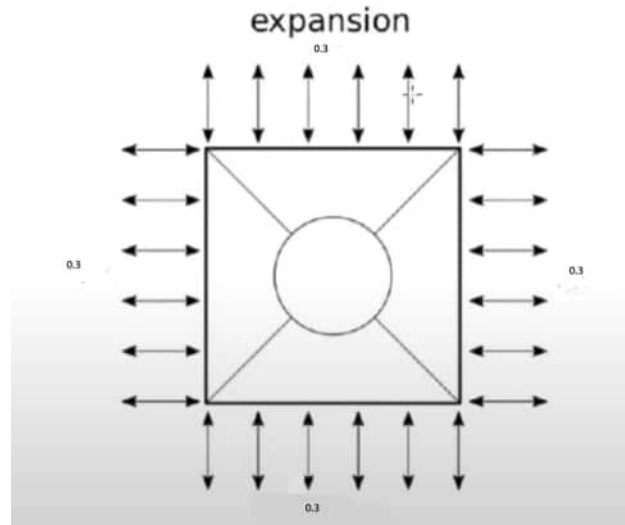


Рис. 3.4:

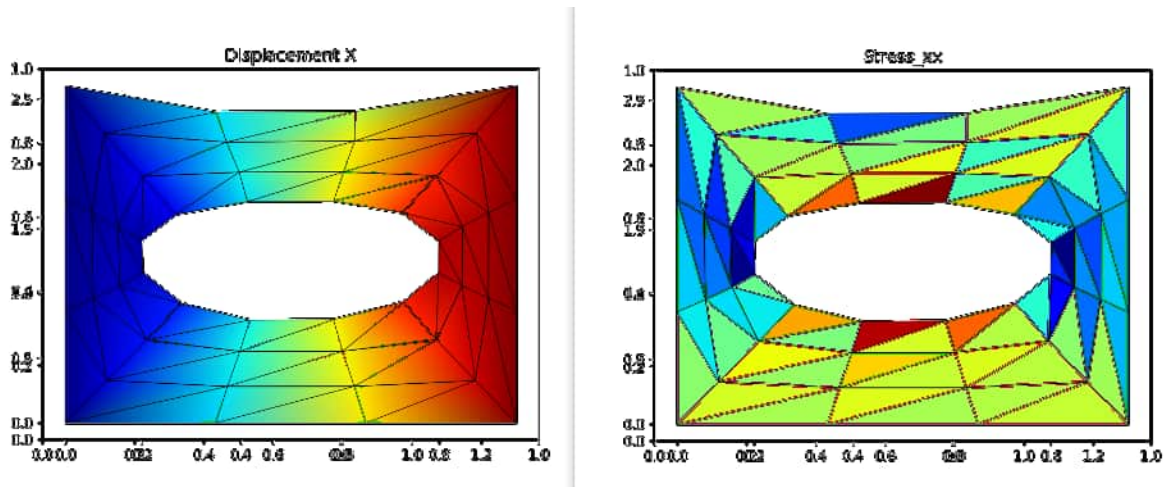


Рис.3.5:

Спробуємо збільшити щільність сітки та отримаємо наступні результати:

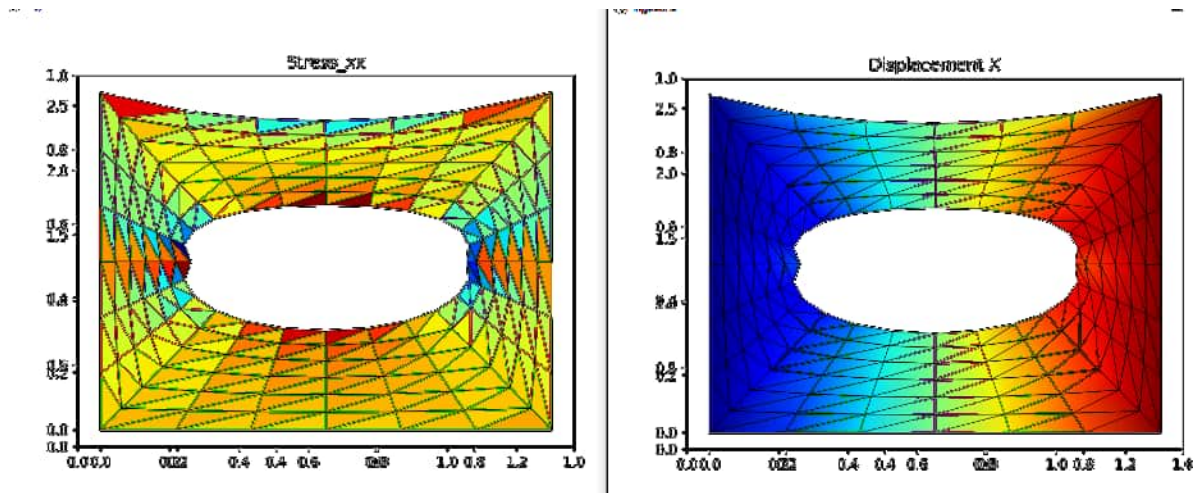


Рис. 3.6:

На основі графіків напружень і деформацій, отриманих з різними параметрами сітки та елементами триангуляції, можна зробити наступні висновки:

1. Висока щільність сітки та використання елементів триангуляції, таких як трикутники, дають більш точні результати; на другому графіку використано вищу щільність сітки та трикутники, що дозволило побачити більш детальні зміни деформацій та напружень.

2. Результати можуть відрізнятися залежно від параметрів сітки та елементів триангуляції. На першому графіку з низькою щільністю сітки і прямокутними елементами видно, що зміни деформацій і напружень є менш детальними і грубими.

3. Збільшення щільності сітки та використання більш точних елементів триангуляції може збільшити обчислювальну складність задачі, але дасть більш точні результати.

4. Аналіз графіків може допомогти виявити області високих напружень або деформацій, які можуть допомогти зрозуміти поведінку матеріалів під дією сил розтягування. Наприклад, можна виявити області концентрованого напруження або деформації, які можуть бути потенційно проблематичними.

Таким чином, метод скінченних елементів може надати значну кількість

інформації про поведінку матеріалів при заданих умовах і геометрії розтягнення.

Висновок

У результаті проведеного дослідження та розробки програмного інтерфейсу на основі методу скінченних елементів було отримано цінні пізнавальні висновки, що сприяють розумінню поведінки матеріалів та розробці ефективних інженерних рішень.

Аналіз графіків напружень і деформацій, отриманих з різними параметрами сітки та елементами триангуляції, дозволив виявити вплив цих факторів на точність і деталізацію аналізу. Використання більш точних параметрів та вищої щільності сітки призвело до отримання більш детальних результатів, що підкреслює важливість вибору оптимальних параметрів для досягнення точних і достовірних висновків.

Крім того, виявлення областей високих напружень і деформацій надало можливість виявити потенційно проблематичні області, що мають практичне значення для розробки стратегій управління напруженнями та деформаціями. Це дозволяє інженерам приймати обґрунтовані рішення щодо зміцнення та оптимізації конструкцій з метою забезпечення безпеки та ефективності.

Реалізація абстракцій у програмному забезпеченні на основі методу скінченних елементів з використанням абстрактних класів з бібліотеки ‘abc’ сприяла покращенню організації коду та забезпечила поліморфізм типів. Це дозволяє розробникам ефективно використовувати програмне забезпечення на основі методу скінченних елементів та зручно взаємодіяти з ним, забезпечуючи гнучкість та розширення системи.

Отже, розроблений програмний інтерфейс на основі методу скінченних елементів разом з виявленими під час дослідження висновками є цінними внесками в галузь інженерних розрахунків. Вони сприятимуть подальшому розвитку методу скінченних елементів та поліпшенню аналізу поведінки матеріалів у різних умовах і конфігураціях, що має важливе значення для розробки безпечних, надійних та оптимальних інженерних рішень.

Список використаної літератури

- [1] Zienkiewicz O., Taylor R., Zhu J. *The Finite Element Method: Its Basis and Fundamentals*.- Sixth edition- Butterworth-Heinemann, 2005. - 802 с.
- [2] Hutton D. *Fundamentals of finite element analysis*.- McGraw-Hill, 2004. - 505 с.
- [3] Голубєва К.М., Денисов С.В., Кашпур О.Ф., Ключин Д.А., Риженко А.І. *Чисельні методи інтегрування (для студентів факультету комп'ютерних наук та кібернетики, ОП «Інформатика»): Методичні розробки.* – Київ: «Видавництво Людмила», 2019. – 55 с.
- [4] <https://silo.tips/download/quadrature-formulas-in-two-dimensions-math-finite-element-method-section-001-spr> (дата звернення: 10.05.2023)
- [5] <https://www.scaler.com/topics/interface-in-python/>(дата звернення: 01.06.2023)

Додатки

Додаток А. Програмна реалізація

Програмну реалізацію цієї дипломної роботи можна знайти на GitHub за цим посиланням:

<https://github.com/kpm-lnu/student-applications/tree/develop/2022-2023/PMP-42/coursework/Iryna/20Kurnytska>