КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет комп'ютерних наук та кібернетики

кафедра математичної інформатики

Кваліфікаційна робота

За спеціальністю 122 Комп'ютерні науки

на тему:

Онлайн калькулятор обчислень методом скінченних елементів

Виконав студент 4-го курсу

Антон КОРОБЕНКО

(підпис)

Науковий керівник:

професор, кандидат фіз.-мат. наук

Ірина ВЕРГУНОВА

Засвідчую, що в цій роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент

(підпис)

Київ – 2022

РЕФЕРАТ

Обсяг роботи: 47 сторінок, 31 ілюстрація, 9 таблиць, 7 джерел посилань.

Ключові слова: метод скінченних елементів, матриця жорсткості, кінцевий елемент, базис, базисна функція, веб сайт, веб застосунок, веб сторінка, онлайн калькулятор.

Об'єктом роботи: метод скінченних елементів та його модифікації, сучасні засоби розробки веб застосунків та веб сторінок.

Мета роботи: розробити онлайн калькулятор з реалізацією методу скінченних елементів для рішення обраної множини класів задач.

Результати роботи: розглянуто особливості застосування методу для різних типів задач та різних степенів апроксимації, реалізований онлайн калькулятор для розв'язання двовимірних задач методом скінченних елементів, виконані тестові розрахунки за допомогою розробленого доданку.

Методи розроблення: методи порівняння, аналізу та синтезу, комп'ютерного моделювання, методи оцінки швидкодії алгоритмів. Інструменти розроблення: безкоштовне, вільно поширюване середовище розробки Visual Studio Code та Spyder IDE, мова програмування Python версії 3.10, JavaScript та CSS, мова розмітки HTML.

Зміст

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧЕННЯ

МСЕ - метод скінченних елементів

СЛАР - система лінійних алгебраїчних рівнянь

КЕ - кінцевий елемент

QU4 (quadrature 4) – кінцевий елемент з 4 вузлами

TR3(triangular 3) – кінцевий елемент з 3 вузлами

PD (problem dimension) – вимірність задачі або ступінь свободи елемента

NoN (number of nodes) – кількість вузлів

NoE (number of elements) – кількість елементів

NPE (nodes per element) – кількість вузлів в 1 елементі

NL (nodes list) – масив вузлів

EL (elements list) – масив елементів

GPE (Gauss points per element) – кількість точок Гауса в елементі

DOFs (degrees of freedom) – ступені свободи

JSON (JavaScript Object Notation) – текстовий формат обміну даних, заснований на JavaScript

REST (Representational State Transfer) – архітектурний стиль взаємодії компонентів розподіленого додатку в мережі

HTML (HyperText Markup Language) – стандартизована мова розмітки документів для перегляду веб сторінок в браузері.

CSS (Cascading Style Sheets) – мова опису зовнішнього вигляду документа, написаного з використанням мови розмітки.

ВСТУП

Оцінка сучасного стану об'єкту дослідження. Сучасні задачі обчислювальної механіки щодня стають все складнішими та більш комплексними, через що часто вирішити їх аналітичним шляхом є неможливим. Єдиним варіантом вирішення подібних задач залишаються чисельні методи, серед яких чільне місце займає метод скінченних елементів. Завдяки своїй універсальності він широко застосовується у різних наукових галузях - біомеханіці, прикладній фізиці, інженерному аналізі, проектуванні, моделюванні та інших.

Основним недоліком МСЕ є велика розмірність системи після збірки, що є проміжним результатом виконання алгоритму, яка потребує спеціальних засобів обробки, зберігання та методів її вирішення.

Актуальність роботи та підстави її виконання. Метод скінченних елементів є сучасним засобом вирішення великої кількості інженерних та фізичних задач. Він дозволяє швидко отримати результат із необхідною точністю для великої кількості задач комп'ютерного моделювання, проте більшість програмного забезпечення, де реалізовано даний метод, є дуже комплексним, доволі складним у використанні і доцільнішим для промислового використання, аніж особистого. Створення онлайн калькулятора надасть простий у використанні ресурс для індивідуальних потреб, а процес виконання буде описувати основні кроки розробки, що може слугувати для створення аналогічних сервісів.

Мета й завдання роботи. Метою роботи є розробка онлайн калькулятора, що реалізує рішення обраної множини двовимірних задач методом скінченних елементів.

Для досягнення мети поставлено наступні завдання:

- проаналізувати принцип роботи та основні прийоми алгоритму;

- розглянути особливості застосування методу для різних типів задач та різних степенів апроксимації;
- розробити та реалізувати веб-додаток для обчислення двовимірних задач;
- виконати тестові розрахунки за допомогою розробленого доданку.

Об'єкт, методи й засоби розроблення. Об’єктом дослідження є процес автоматизації використання методу скінченних елементів та його модифікацій, сучасні засоби розробки веб застосунків.

Використані методи розроблення: методи порівняння, аналізу та синтезу, комп’ютерного моделювання, методи оцінки швидкодії алгоритмів.

Засоби розроблення: безкоштовне, вільно поширюване середовище розробки Visual Studio Code, мова програмування Python версії 3.10, JavaScript та CSS, мова розмітки HTML.

Можливі сфери застосування. Дана робота може знайти практичне застосування у навчальному процесі та для виконання прикладних досліджень користувачами, які не мають доступу до спеціальних інженерних застосунків.

РОЗДІЛ 1. МЕТОД СКІНЧЕННИХ ЕЛЕМЕНТІВ

1.1 Теоретичні засади МСЕ

Метод скінченних елементів - це процедура вирішення задач, які сформовані у вигляді диференціального рівняння або варіаційного принципу. Він успішно використовується для вирішення складних рівнянь пружності, стискання, теплопровідності, стискання та інших механічних задач, є незамінним якщо потрібно враховувати геометричні особливості об'єкта. З математичного боку він є узагальненням класичного метода Релея-Рітца-Гальоркіна, проте відрізняється представленням апроксимуючої функції. В МСЕ вона є лінійною комбінацією неперервних кусково-лінійних функцій. Кожна така функція $\varphi(x)$ дорівнює нулю на майже всій області визначення, проте відмінна від нуля в околі одного вузла.

Фізична постановка задачі найчастіше представляється певною конструкцією з прикладеними до неї силами та точками опори. Математична постановка досягається за допомогою використання певних припущень для ідеалізації фізичної постановки задачі.

Математична модель повинна поєднувати в собі дві основні властивості: ефективність та достовірність. Найефективнішою буде та модель, котра отримає результат із заданою точність, при цьому не перевищивши задані витрати ти для знаходження. Найбільш достовірною вважається та модель, результат якої з певною точністю збігається з результатом обчислення максимально повної моделі (рис. 1.1).

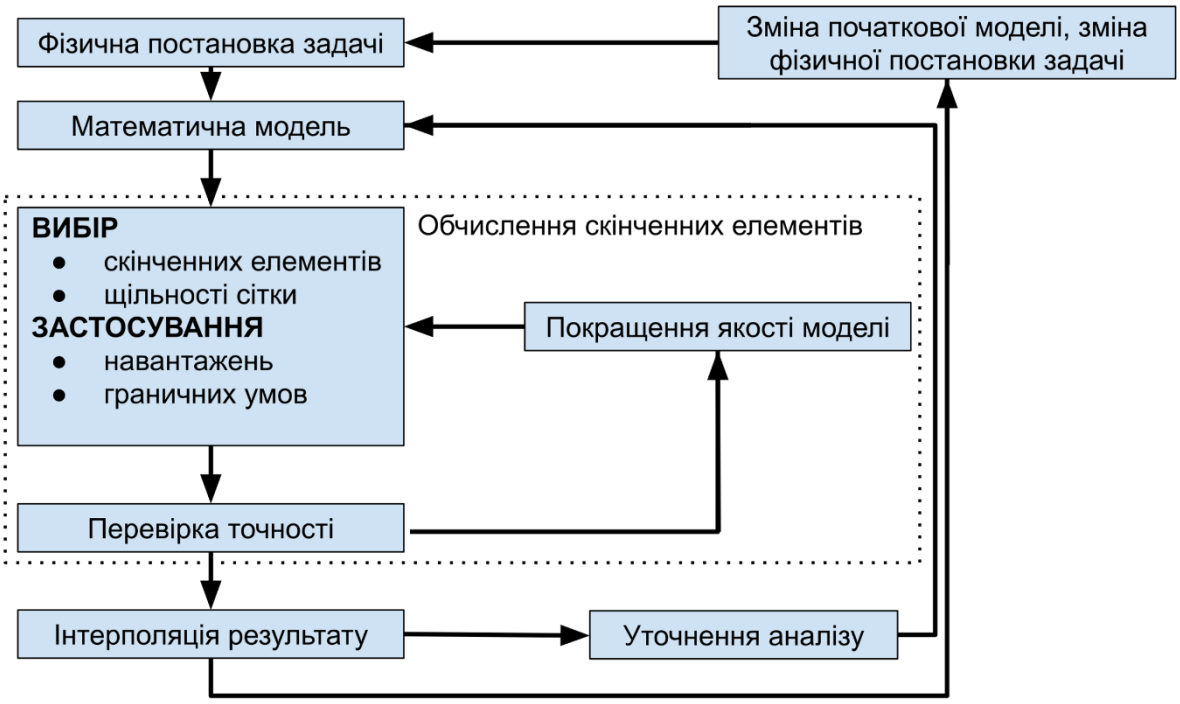


Рисунок 1.1 Принцип використання МСЕ

Джерело: складено автором на основі [1]

Основні кроки методу:

1) Розбиття області та побудова сітки елементів з однаковою структурою. Елементи не мають перетинатись та мають бути просто типу, найчастіше використовують прямокутники або трикутники. Такі елементи називають кінцевими елементами (КЕ). Вершини КЕ називають зовнішніми вузлами, вони потрібні для опису геометрії елемента, з’єднання елементів один з одним і для компонент розв'язку(ступенів свободи). Також КЕ може мати внутрішні вузли, які забезпечують більш точне задання шуканих функцій. Чим більша кількість елементів - тим краща точність, проте зростає час на виконання роботи. В залежності від задачі число ступенів свободи у вузлі може варіювати: в задачах на теплопровідність достатньо

одного ступеня - температури, а в задачах пружності необхідно мінімум два - переміщення по осі X та Y.

2) Наступний крок - вибір апроксимуючої функції в найпростішій формі, найчастіше - поліном, через що простір задачі часто називають простором кусково-поліноміальних функцій.

3) Задання граничних умов та формування СЛАР з урахуванням попередніх результатів [4].

4) Розв’язок СЛАР.

5) Визначення шуканих величин в елементах.

Як я вже писав вище, МСЕ - це об’єднання методів, основаних на проекційних методах вирішення рівнянь або варіаційних методах мінімізації функціоналів.

Розглянемо проекційні методи. Нехай поставлено задачу знайти наближений розв'язок диференційного рівняння

$$F(u) = 0 \quad (1),$$

де оператор $F: E -> H$, де E, H - два нормованих простори, та в цих просторах існують базиси виду: $\{\varphi_n\}, \{\phi_n\}, n = \overline{0, \infty}$, такі, що $\forall x \in E, y \in H$ знайдуться послідовності $\{a_n\}, \{b_n\}$, такі, що $x = \sum_{n=1}^{\infty} a_n \varphi_n , , y = \sum_{n=1}^{\infty} b_n \phi_n$. Через P_n, Q_n позначимо оператори проектування на простори E_n, H_n перших n елментів: $E_n = P_n E$, $H_n = Q_n H$. Суть проекційних методів полягає в заміні рівняння (1) наближеним рівнянням

$$F_n(\bar{u}) = 0 \quad (2),$$

де $\bar{u} \in \mathrm{E}_n, \mathrm{F}_n: \mathrm{E}_n \to \mathrm{H}_n, \mathrm{F}_n = \mathrm{Q}_n F$. Вибір різних базисів φ_n, ϕ_n і різних проекторів приводить до різних проекційних методів, але в цій роботі вони не будуть описані.

Проте я опишу загальну схему метода Гальоркіна. В ньому простір E_n та простір H_n збігаються, тобто $\varphi_n = \phi_n$. Цей метод будує наближений розв'язок рівняння $F(u) = f - Lu = 0$ або $Lu = f$ за наступними кроками:

1) Обирається базис $\varphi_i, i = \overline{1,n}$.

2) Наближений розв'язок шукаємо у вигляді $\bar{u} = \sum_{i=0}^{n} \mathrm{a}_i \varphi_i$.

3) Коефіцієнти a_i визначаються з розв'язку СЛАР, отриманої з умови ортогональності $F(\bar{u}) = f - L\bar{u}$ до $\varphi_1, \ldots, \varphi_n$.

Варіаційні принципи полягають у відшуканні функції, що задає певному функціоналу найбільше або найменше значення. Зазвичай такий принцип формується у вигляді задачі мінімізації інтегралу деякої функції.

Розглянемо метод пошуку наближеного розв'язку задачі мінімізації методом Рітца. Ідея полягає в тому, щоб шукати мінімум функціонала на просторі H_n. Елементи $\nu \in \mathrm{H}_n$ називаються пробними функціями. Апроксимацією Рітца називають функцію $u^h \in \mathrm{H}_n$, яка мінімізує F на підпросторі H_n: $F(u^h) \leq F(\nu), \forall\, \nu \in \mathrm{H}_n$.

Основні кроки:

1) Вибір підпростору H_n і базису $\varphi_i, i = \overline{1,n}$.
2) Наближений розв'язок має вигляд $u^h = \sum_{i=0}^{n} \mathrm{a}_i \varphi_i$.

3) Коефіцієнти a_i і можна знайти з умов мінімізації функціонала $F(u^h)$ за параметрами a_i, які дадуть СЛАР: $\frac{\partial F(u^h)}{\partial a_i} = 0,, i = \overline{1,n}$.

1.2 Дискретизація та простір скінченних елементів

Розбиття простору на менші елементи (дискретизація) включає в себе процеси задання числа, розміру та форми підпросторів (елементів) які будуть застосовуватись для створення дискретної моделі фізичного об’єкта. Як вже було сказано, більша кількість елементів буде позитивно впливати на точність результату, проте може значно уповільнити час виконання методу. Найпростішим елементом, який може використовуватись - одновимірний елемент. Зазвичай це відрізок, проте він може мати поперечний розріз (зазвичай використовується для задач з стержневими конструкціями).

Процес дискретизації зазвичай розбивається на два етапи - розбиття об’єкта та нумерацію вузлів та елементів. Детальніше розглянемо процес на прикладі найчастішого розбиття: тріангуляції - розбиття на трикутні елементи. Найпростіший спосіб розбити площину на трикутники - вибрати деяку кількість вузлів на кожній стороні, з'єднати відповідні вузли прямими, а точки перетину вважати новими вузлами. На практиці частіше використовують спеціалізовані алгоритми тріангуляції площин, оскільки це значно ефективніше, швидше та задає більш рівномірне розбиття. В деяких випадках необхідно штучно збільшити гущину елементів для підвищення точності результатів, наприклад при різких змінах геометрії фізичного об’єкта.

Хоча нумерація елементів може здатись тривіальною частиною дискретизації в МСЕ, це не так. При застосуванні методу доводиться вирішувати СЛАР, в якій багато коефіцієнтів є рівними нулю. При більш детальному розгляді матриці коефіцієнтів (рис. 1.2) можна помітити, що всі ненульові коефіцієнти, як і деякі нульові, будуть розташовуватись між двома лініями, паралельними головній діагоналі. Відстань між цими лініями називають шириною полоси матриці, всі коефіцієнти поза цією смугою дорівнюють нулю, тому їх можна не зберігати в пам’яті.

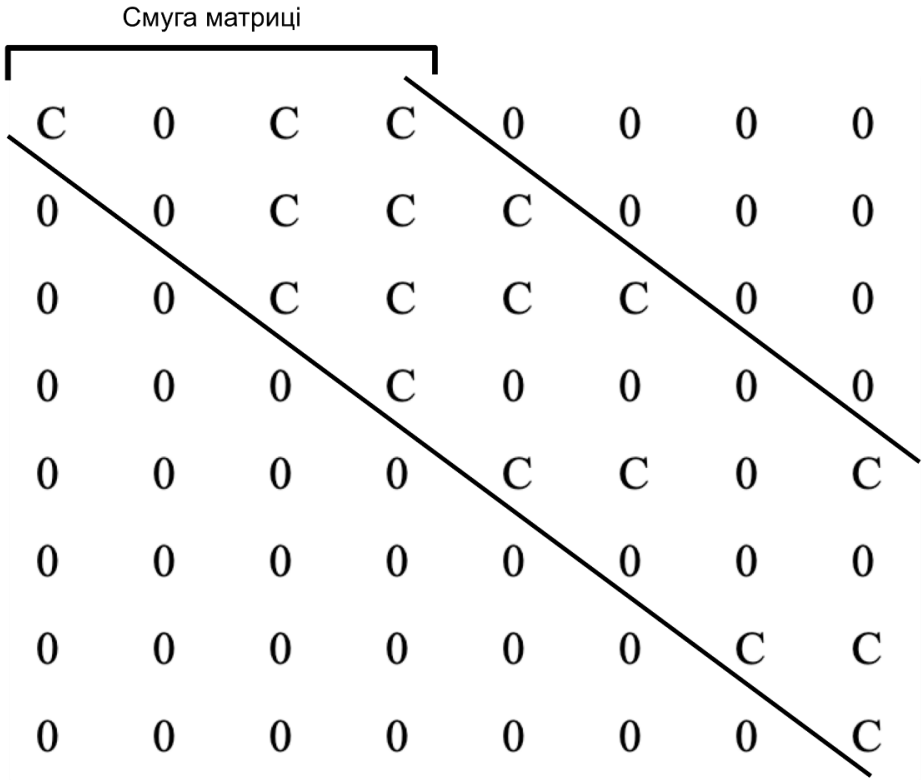


Рисунок 1.2 Ширина смуги матриці

Джерело: складено автором на основі [1]

Формула для обчислення ширини смуги: $B = (R + 1) * Q$, де R - найбільша різниця номерів вузлів в елементі, Q - число ступенів свободи в

кожному вузлі. Якщо необхідно мінімізувати розмір B, то потрібно мінімізувати Q шляхом послідовної нумерації вузлів при русі до мінімального розміру об'єкта. Оптимальна нумерація може пришвидшити час виконання на 50%.

Найпростіший одновимірний елемент має два вузли, проте одновимірні елементи вищих порядків можуть мати більше в залежності від об'єктів, які вони мають описувати (рис. 1.3).

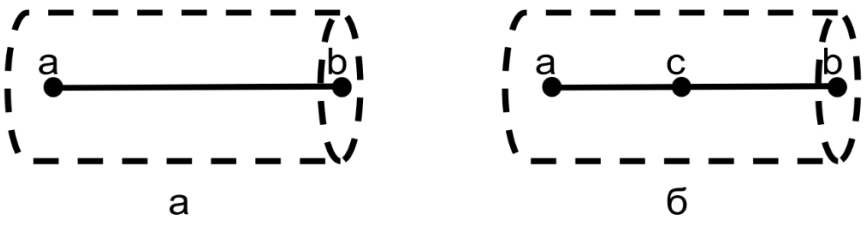


Рисунок 1.3 Одновимірні елементи

(а - лінійний, б - квадратичний)

Розглянемо як визначаються кусково-елементні базисні функції (також називають функціями форми) при розв’язку одновимірної задачі на відрізку $V = [a, b]$. Спочатку даний відрізок потрібно розбити на n кінцевих елементів $e_k = (x_k, x_{k+1})$, $k = \overline{1, n}$, довжина сітки $h = x_{k+1} - x_k$ (за умови що всі елементи однакові), Φ_i - значення у і-му вузлі. Головними властивостями функцій форм є:

1) функція N_i дорівнює одиниці лише у вузлу хі, та нулю у решті вузлів;

2) функція N_i відмінна від нуля лише в елементах, які містять вузол x_i.

Запишемо поліном для скалярної величини: $\varphi = a_1 + a_2 x$.

Константи a_1 та a_2 визначаються з умов у вузлах елемента: $\varphi = \Phi_i$ при $x = \mathrm{x}_i$ та $\varphi = \Phi_j$ при $x = \mathrm{x}_j$. Маємо систему рівнянь для знаходження констант: $\begin{cases} a_1 + a_2 \mathrm{x}_i = \Phi_i \\ a_1 + a_2 \mathrm{x}_j = \Phi_j \end{cases}$.

Розв'язавши систему, отримаємо: $a_1 = \frac{\Phi_i \mathrm{x}_j - \Phi_j \mathrm{x}_i}{h}, a_2 = \frac{\Phi_j - \Phi_i}{h}$.

Підставивши отримані результати до початкового поліному отримаємо: $\varphi = \frac{\Phi_i \mathrm{x}_j - \Phi_j \mathrm{x}_i}{h} + \left(\frac{\Phi_j - \Phi_i}{h}\right) x$, що можна переписати як

$$\varphi = \frac{\mathrm{x}_j - x}{h} \Phi_i + \left(\frac{x - \mathrm{x}_i}{h}\right) \Phi_j \qquad (3).$$

Лінійні функції, що залежать від x і є функціями форми. Їх позначають як N_i, де i - індекс вузла.

Лінійний трикутний елемент задається трьома вузлами-вершинами i, j, k, які утворюють трикутник з прямими сторонами [5][6]. Аналогічно одновимірному елементу, розглянемо базисні функції для трикутного елемента (рис. 1.4, рис. 1.5).

При роботі з вузлами необхідно притримуватись нумерації або певного напрямку їх обходу в елементі. Частіше використовують напрямок проти годинникової стрілки. Вузли будемо позначати через i, j, k, а координати вузлів - $(X_i, Y_i), (X_j, Y_j), (X_k, Y_k)$.

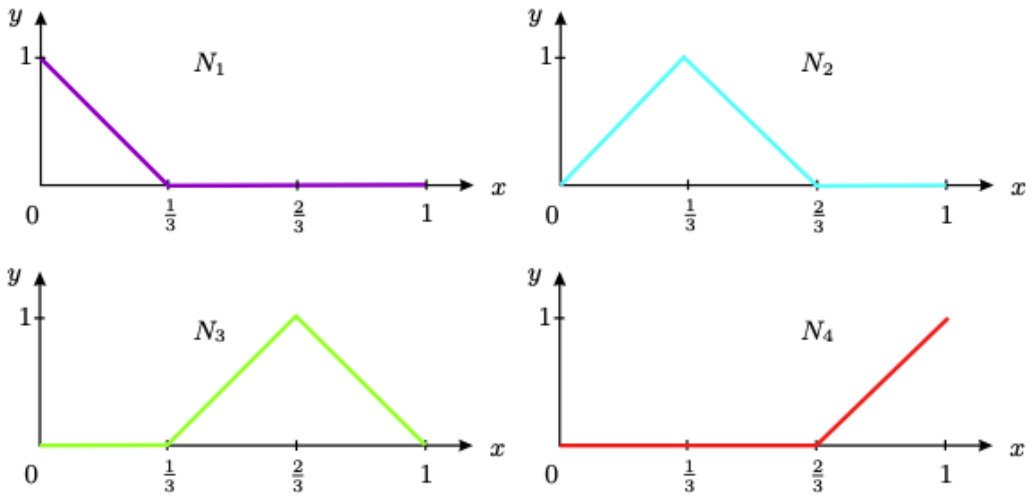


Рисунок 1.4 Базисні функції N_i для трьохелементної області

Джерело: складено автором на основі [2][3]

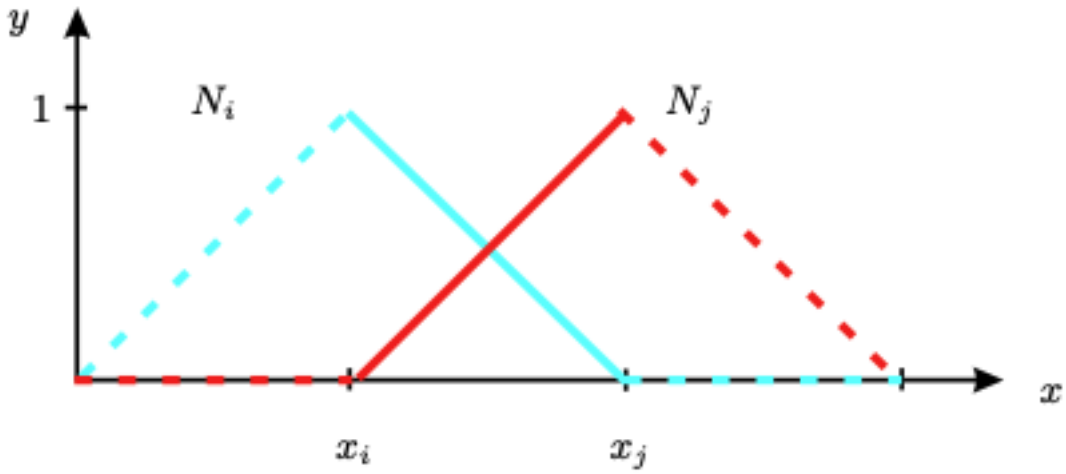


Рисунок 1.5 Базисні функції N_i, N_j елемента $[x_i, x_j]$

Джерело: складено автором на основі [2][3]

Інтерполяційний багаточлен: $\varphi = a_1 + a_2 x + a_3 y$ [6].

Умови для вузлів: $\varphi = \Phi_i$ при $x = X_i$ та $y = Y_i$, $\varphi = \Phi_j$ при $x = X_j$ та $y = Y_j$, $\varphi = \Phi_k$ при $x = X_k$ та $y = Y_k$. Підставивши задані умови до виразу (3) отримаємо СЛАР з трьох рівнянь для знаходження a_1, a_2, a_3. Розв'язавши систему, отримаємо:

$$a_1 = (1/2A) * ((X_j Y_k - X_k Y_j)\Phi_i + (X_k Y_i - X_i Y_k)\Phi_j + (X_i Y_j - X_j Y_i)\Phi_k),$$

$$a_2 = (1/2A) * ((Y_j - Y_k)\Phi_i + (Y_k - Y_i)\Phi_j + (Y_i - Y_j)\Phi_k),$$

$$a_3 = (1/2A) * ((X_k - X_j)\Phi_i + (X_i k-)\Phi_j + (X_j - X_i)\Phi_k),$$

A – площа елемента, обчислюється за формулою: $A = \frac{1}{2}\begin{vmatrix} 1 & X_i & Y_i \\ 1 & X_j & Y_j \\ 1 & X_k & Y_k \end{vmatrix}$.

Підставивши отримані значення коефіцієнтів в початковий многочлен, отримаємо вираз, з якого можемо знайти N_i.

Також для дискретизації складніших об'єктів можуть використовуватись складніші форми кінцевих елементів, такі як тетраедри та гексаедри. В деяких випадках елементи можуть мати криволінійні грані, що може покращити точність. В цій роботі не будуть описуватись такі структури, проте вони є аналогічними до тих, що розглядались вище. Також існує достатньо матеріалів у вільному доступі для поглибленого вивчення подібних модифікацій методу.

1.3 Застосування та реалізація МСЕ

Розглянемо реалізацію МСЕ для одновимірного випадку [4]. Вхідні дані: n – кількість вузлів, $m = n - 1$ – кількість елементів, x – координати вузлів $x_i, i = \overline{1,n}$, $k = (k_{ij})(2x2)$ – матриця елемента, вектор правих частин елементів - $b = (b_i), i = \overline{1,n}$, глобальна матриця жорсткості – $K = (a_{ij})$, розміру $n \times n$, вектор результатів - $u = (u_i)$, $i = \overline{1,n}$, вектор правих частин - $f = (f_i)$, $i = \overline{1,n}$, c, d – координати початку і кінця відрізка, на якому ми розглядаємо рівняння, $u_s = t$ – гранична умова.

Алгоритм:

1) Знаходимо координати вузлів. Крок сітки: $h = \frac{(d-c)}{m}$. $for\ (i = 0; i < n; i = i + 1)$: $x[i] = c + (i - 1)h$.

2) $for\ (i = 0; i < m; i = i + 1)$:

1. Формуємо матрицю k елемента.
2. Формуємо глобальну матрицю жорсткості K:

$$a_{ii} = a_{ii} + k_{ii}$$

$$a_{ii+1} = a_{ii+1} + k_{12}$$

$$a_{i+1i} = a_{i+1i} + k_{21},$$

$$a_{i+1i+1} = a_{i+1i+1} + k_{22}.$$

3. Формуємо праву частину елементного вектору.
4. Формуємо глобальний вектор правих частин

$f_i = f_i + b_1,\ \ f_{i+1} = f_{i+1} + b_2.$

3) Вносимо граничну умову $u_s = t$ в матрицю і вектор правих частин

1. $a_{ss} = 1, f_s = t$
2. Для $k = 1, \dots, n\ \&\ k \neq \{$

$a_{sk} = 0;$

$f_k = f_k - a_{ks} * t;$

$a_{ks} = 0;\}$

4) Розв’язуємо систему $Au = f$

5) Повертаємо результат.

Розглянемо програмну реалізацію для розв’язку задачі теплопровідності методом скінченних елементів. Використаємо трикутні елементи для дискретизації фізичного об’єкта. Для наочності розглянемо елемент об’єкта (рис. 1.6) та таблицю елементів (таблиця 1.1).

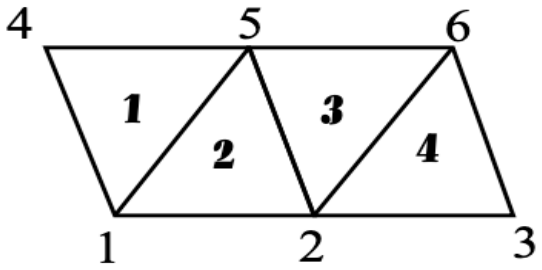


Рисунок 1.6 Приклад розбиття частини об’єкта

Джерело: складено автором на основі [2]

Номер	Вузол i	Вузол j	Вузол k
1	1	5	4
2	1	2	5
3	2	6	5
4	2	3	6

Таблиця 1.1 Список вузлів, з яких складаються елементи

Джерело: складено автором на основі [2]

Перейдемо до програмної реалізації. Вхідні дані: n – кількість вузлів, m – кількість елементів, масив вузлів $grid[n][2]$: $grid[i][0] -$

координата $x, grid[i][1] -$ координата y, масив елементів $elements[m][3]: elements[i] = [node1, node2, node3]$, вектор правих частин елементів - $b = (b_i), i = \overline{1,3}$, $k = \left(k_{ij}\right)(3x3)$ – матриця елемента, вектор результатів - $u = (u_i)$, $i = \overline{1,n}$, вектор правих частин - $f = (f_i)$, $i = \overline{1,n}$, масив властивостей матеріалу для кожного елемента - $material[m][2]: material[i] = [0.3, 2000]$, $u_s = t$ – гранична умова.

Алгоритм:

1) Зчитуємо вхідні дані та формуємо необхідні масиви.
2) Для кожного елемента в elements:

1. Визначаємо номери вузлів кожного елемента
   $for(i = 0; i < 3; i++): \{s_j = elements[j][i]\}$
   $s_1, s_2, s_3 -$ вузли елемента j.
2. Формуємо елементну матрицю.
3. Формуємо глобальну матрицю жорсткості
   $for(j = 0; j < 3; j++):$
   $for\ (i = 0; i < 3; i++):$
   $\{a_{s_j,s_i} = a_{s_j,s_i} + k_{j,i}\}$
4. Формуємо вектор правих частин для елементів.
5. Формуємо глобальний вектор правих частин
   $for(i = 0; i < 3; i++): \{f_{s_j} = f_{s_j} + b_j\}$

3) Вносимо граничні умови в матрицю і вектор правих частин.
4) Розв’язуємо систему $Ku = f$.
5) Виводимо результат u.

1.4 Реалізація МСЕ на комп’ютері

При проектуванні методу скінченних елементів на ЕОМ виникає проблема зберігання та обробки глобальної матриці та елементної матриць жорсткості. Це пояснюється тим, що при стандартному підході вони містять велику кількість елементів рівних нулю, до того ж матриці жорсткості елементів мають таку саму розмірність, що і глобальна матриця жорсткості. Також глобальна матриця формується шляхом додавання елементних матриць, що вимагає їхнього зберігання в пам’яті.

Ефективні моделі методу передбачають використання скорочених форм запису матриць елементів, що значно зменшує кількість необхідної пам’яті. Розглянемо такий підхід детальніше.

Розглянемо окремий елемент та його матрицю. Приберемо всі ступені свободи, які не стосуються даного елемента. Функції форми запишемо відповідно до порядку вузлів. Розглянемо третій елемент з рис. 1.6 (з урахуванням рис. 1.7).

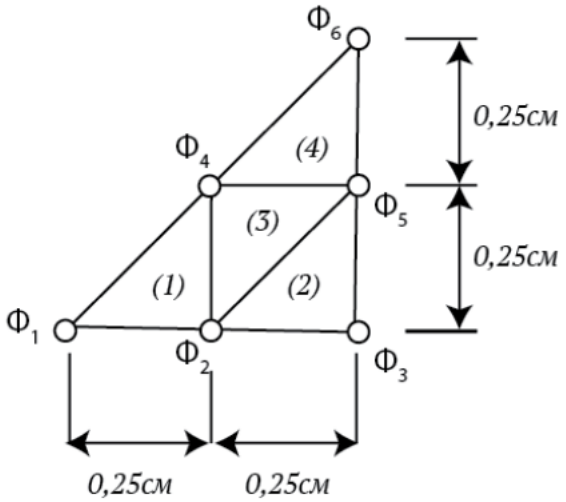


Рисунок 1.7 Область V, розбита на 4 елементи

Джерело: складено автором на основі [1]

Функція форми: $\varphi = N_2\Phi_2 + N_4\Phi_4 + N_5\Phi_5$ (Φ_1, Φ_3 та Φ_6 множать на 0, адже вони не стосуються третього елемента), якщо впорядкуємо функції форми з вузла i проти годинникової стрілки, отримаємо: $\varphi = N_2\Phi_2 + N_5\Phi_5 + N_4\Phi_4$.

Матриця градієнтів має наступний вигляд:

$$\{g\} = \begin{Bmatrix} \frac{\partial\varphi}{\partial x} \\ \frac{\partial\varphi}{\partial y} \end{Bmatrix} = \frac{1}{2A^3}\begin{bmatrix} b_2^3 & b_5^3 & b_4^3 \\ c_2^1 & c_5^1 & c_4^1 \end{bmatrix}\begin{Bmatrix} \Phi_2 \\ \Phi_5 \\ \Phi_4 \end{Bmatrix} = [B^3]\{\Phi^3\}.$$

Коефіцієнти b і c знаходяться з співвідношень для функцій форми, описаних вище. Після цього їхнє значення підставляється до $[B^3]$, в даному прикладі отримаємо:

$$\{g\} = \begin{bmatrix} 0 & 4 & -4 \\ -4 & 0 & 4 \end{bmatrix}\begin{Bmatrix} \Phi_2 \\ \Phi_5 \\ \Phi_4 \end{Bmatrix}.$$

Підставимо отримані значення в форму знаходження матриці жорсткості елемента: $[k] = \int [B^3]^T [B^3] dV$, отримаємо

$$[k] = \begin{bmatrix} 0{,}5 & 0 & -0{,}5 \\ 0 & 0{,}5 & -0{,}5 \\ -0{,}5 & -0{,}5 & 1 \end{bmatrix}$$

Таким чином ми маємо квадратну матрицю розмірністю n = 3 замість n = 6 при стандартному підході, тобто зменшили об’єм зайнятої пам’яті ЕОМ в ~ 4 рази.

РОЗДІЛ 2. РОЗРОБКА ОНЛАЙН КАЛЬКУЛЯТОРА

2.1 Підготовка до розробки

Перед початком роботи потрібно визначити тип задач, які буде розв'язувати майбутній калькулятор. У моєму випадку це будуть двовимірні задачі, ступені свободи елементів будуть дорівнювати 2. У роботі будуть розглядатися лише прості об'єкти, а саме прямокутні. Користувачеві необхідно ввести параметри розміру об'єкту (висота та ширина), його властивості, кількість кінцевих елементів по осі X та Y, а також їхній тип.

Онлайн версія програми включає в себе застосування останніх технологій створення веб додатків та архітектурні рішення. Застосунок повинен бути розробленим з оглядом на майбутнє розширення функціоналу та легку масштабованість.

Для зручного використання програми також необхідно розробити інтуїтивний та простий інтерфейс взаємодії з користувачем.

2.2 Дискретизація та створення елементів

Згідно з алгоритмом МСЕ, першим кроком має бути дискретизація фізичного об'єкта на кінцеві елементи. У даному випадку розглянемо два варіанти розбиття на елементи з чотирма та трьома вузлами (рис. 2.1, рис. 2.2). Для цього створимо функцію **generate_uniform_2d_mesh()**, яка буде приймати наступні аргументи: **width** - ширину, **height** - висоту, **elements_on_X** - кількість елементів по осі X, **elements_on_Y** - кількість елементів по осі Y, **element_type** - тип елементів.

Початкова умова: нульовий вузол об’єкта розташовується в точці (0,1) на системі координат, розрахунок позицій решти вузлів засновується на положенні нульового. Наприклад, крайній верхній правий вузол при вхідних параметрах width = n, height = m матиме координати (n, m+1).

Складемо два розбиття об’єкта: на чотирикутники (рис 2.1) та на трикутники (рис 2.2), та пронумеруємо номери вхідних вузлів чорним кольором, а номери отриманих елементів – червоним.

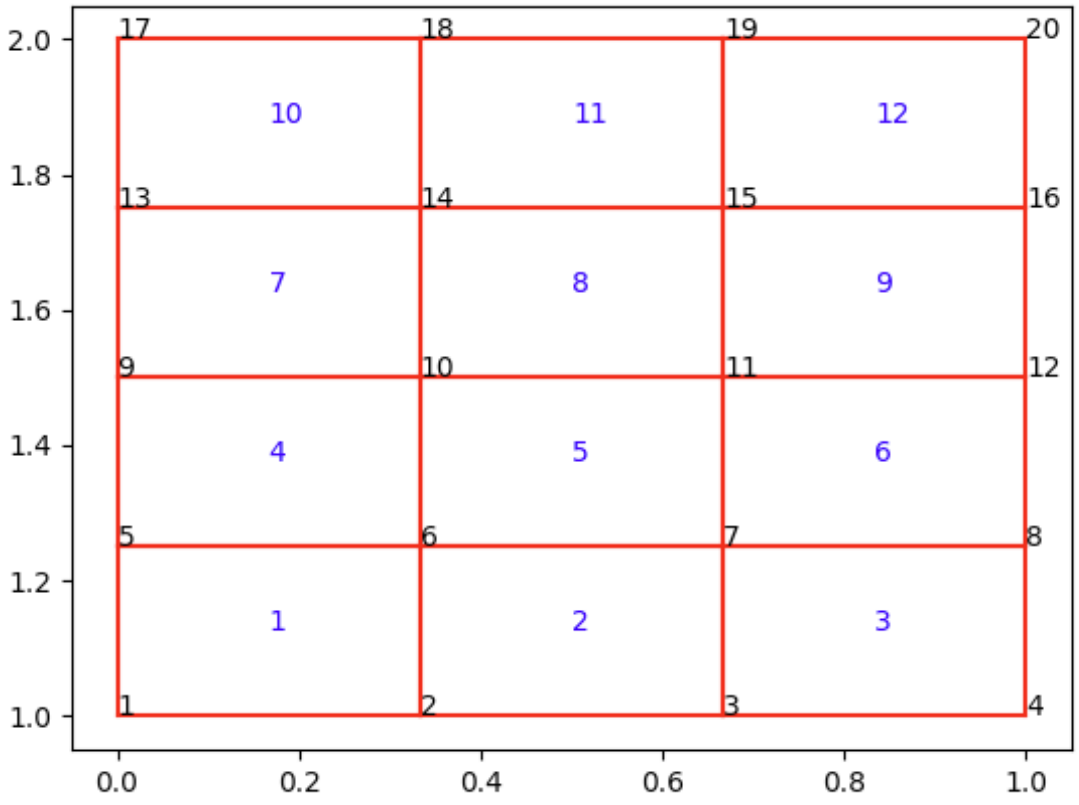


Рисунок 2.1. Результат розбиття на елементи з 4 вузлами

Джерело: складено автором

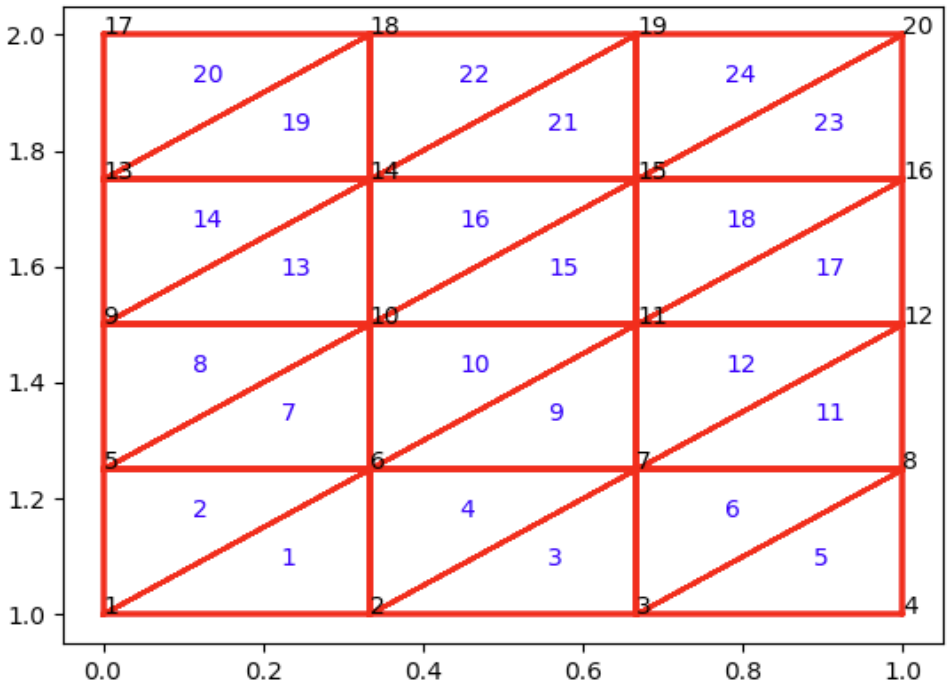


Рисунок 2.2. Результат розбиття на елементи з 3 вузлами

Джерело: складено автором

Розгляньмо бажані результати роботи алгоритмів для наступних вхідних даних: **width** – 1 см, **height** – 1 см, **elements_on_X** – 3, **elements_on_Y** – 4, **element_type** – ‘QU4’ та ‘TR3’ відповідно.

Надалі для скорочення зробимо заміну: $n = elements\_on\_X, m = elements\_on\_Y$.

Для створення сітки елементів перш за все необхідно дізнатись вимірність елемента по осі X та Y. Задля цього використаємо наступні формули:

$$step\_x = width/n, \quad step\_y = height/m.$$

Якщо розглядати n та m як кількість стовбців та рядків сітки елементів, то можна дізнатись загальну кількість елементів: $NoE = n * m$, та сформувати наступний алгоритм генерації вузлів:

Для кожного $i \in \overline{0,n}$ та для кожного $j \in \overline{0,m}$:

$$\text{NL.append}\ (i * step\_x, i * step\_y).$$

Отримавши заповнений масив вузлів, індекс його елемента відповідатиме індексу вузла на рисунку. Тепер можемо перейти до процесу генерації елементів. Опишемо логіку алгоритму для розбиття на чотирикутники.

Для кожного $i \in \overline{0,n}$ та для кожного $j \in \overline{0,m}$:

якщо елемент знаходиться на лівому ребрі об'єкту, тоді:

$$EL_{i*n+j,0} = i * (n+1) + j\ ,$$

$$EL_{i*n+j,1} = EL_{i*n+j,1} + 1,$$

$$EL_{i*n+j,3} = EL_{i*n+j,1} + n + 1,$$

$$EL_{i*n+j,2} = EL_{i*n+j,3} + 1.$$

якщо елемент не на лівому ребрі:

$$EL_{i*n+j,0} = EL_{i*n+j-1,1}\ ,$$

$$EL_{i*n+j,3} = EL_{i*n+j-1,2},$$

$$EL_{i*n+j,1} = EL_{i*n+j,0} + 1,$$

$$EL_{i*n+j,2} = EL_{i*n+j,3} + 1.$$

Маємо масив EL заповнений чотирма індексами вузлів для кожного елемента. Отже, ми виконали розбиття об'єкта на чотирикутники. Щоб виконати тріангуляцію, необхідно ініціалізувати та заповнити новий EL, кожен елемент якого цього разу буде зберігати лише три вузли. NoE_t буде вдвічі більшою, адже кожен чотирикутник поділиться на дві частини. Опишемо логіку цього алгоритму.

$$NoE_t = 2 * NoE;\ \ EL_t = [0] * NoE_t.$$

Для кожного $i\ \in \overline{0, \mathrm{NoE}}$:

Так як чотирикутник ділиться на два трикутники, то розглядається генерація кожного з них:

- Для першого:

$$EL_{t[2*\mathrm{i},0]} = EL[i, 0],$$
$$EL_{t[2*\mathrm{i},1]} = EL[i, 1],$$
$$EL_{t[2*\mathrm{i},1]} = EL[i, 2].$$

- Для другого:

$$EL_{t[2*i+1,0]} = EL[i, 0],$$
$$EL_{t[2*i+1,1]} = EL[i, 2],$$
$$EL_{t[2*i+1,2]} = EL[i, 3].$$

Тепер, заповнивши EL_t, ми повність виконали перший крок методу скінченних елементів. Використовувати можна обидва методи розбиття, проте тріангуляція точніше описує геометрію об’єкта, в результаті чого результат буде точнішим, проте такий підхід займе більше часу та пам’яті: лише для зберігання трикутних елементів знадобиться майже вдвічі більше пам’яті.

2.3 Розрахунок матриці жорсткості елемента

Для розрахунку матриці жорсткості елемента необхідно ввести нове поняття – точка Гауса, яку ще називають точкою інтегрування, оскільки числове інтегрування здійснюється саме в них [7]. Для того щоб отримати шукану матрицю та компоненти для інших матриць, програма реалізація МСЕ повинна використовувати чисельне інтегрування за площею (або

об’ємом) об’єкта. Також точки Гауса використовуються після знаходження переміщень у вузлах, оскільки напруги визначаються найточніше саме в цих точках. Елементи можуть мати різну кількість точок Гауса: ті, які містять лише одну називають елементами зменшеного інтегрування, вони швидше обробляють програмою, проте в деяких випадках можуть неправильно представляти геометрію фігури.

Правила точок Гауса для трикутних елементів:

$$\int_0^1 \int_0^{1-\eta} \{\blacksquare\} d\xi d\,\eta \approx \frac{1}{2} * \sum_{i=1}^{n} \alpha_i \{\blacksquare\}|_{gp_i}, \qquad (4)$$

де $n -$ к $-$ ть точок Гауса в елементі, $gp_i -$ і $-$ та точка Гауса

З рівняння (4) можемо вивести правила точок Гауса в трикутниках (таблиця 2.1, таблиця 2.2).

Номер точки Гауса	Координати		Фактор ваги
	ξ	η	α
1	1/3	1/3	1

Таблиця 2.1 Правило однієї точки Гауса в трикутнику

Номер точки Гауса	Координати		Фактор ваги
	ξ	η	α
1	1/6	1/6	1/3
2	4/6	1/6	1/3
3	1/6	4/6	1/3

Таблиця 2.2 Правило трьох точок Гауса в трикутнику

Правило точок Гауса для чотирикутних елементів:

$$\int_{-1}^{1}\int_{-1}^{1}\{\blacksquare\}d\xi d\,\eta \approx* \sum_{i=1}^{n}\alpha_i\{\blacksquare\}|_{gp_i}, \quad (5)$$

де $n -$ кількість точок Гауса в елементі, $gp_i -$ i-та точка Гауса.

Алогічно з (5) можемо вивести правила точок Гауса чотирикутниках (таблиця 2.3, таблиця 2.4).

Номер точки Гауса	Координати		Фактор ваги
	ξ	η	α
1	0	0	2*2

Таблиця 2.3 Правило однієї точки Гауса в чотирикутнику

Номер точки Гауса	Координати		Фактор ваги
	ξ	η	α
1	$-1/\sqrt{3}$	$-1/\sqrt{3}$	1*1
2	$1/\sqrt{3}$	$-1/\sqrt{3}$	1*1
3	$1/\sqrt{3}$	$1/\sqrt{3}$	1*1
4	$-1/\sqrt{3}$	$1/\sqrt{3}$	1*1

Таблиця 2.4 Правило чотирьох точок Гауса в чотирикутнику

Також у процесі виконання етапу доведеться матрицю Якобі. Вона використовується при переході з натуральної системи координат в глобальну(фізичну).

$$J_{[2x2]} = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 \\ y_1 & y_2 & y_3 & y_4 \end{bmatrix} \begin{bmatrix} N_\xi^1 & N_\eta^1 \\ N_\xi^2 & N_\eta^2 \\ N_\xi^3 & N_\eta^3 \\ N_\xi^4 & N_\eta^4 \end{bmatrix}, \text{ або } J_{[2x2]} = \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} \end{bmatrix}$$

Ініціалізуємо $grad\_nat = \begin{bmatrix} N_\xi^1 & N_\xi^2 & N_\xi^3 & N_\xi^4 \\ N_\eta^1 & N_\eta^2 & N_\eta^3 & N_\eta^4 \end{bmatrix}$, виникає питання звідки взяти значення $N_\xi^1, \dots, N_\eta^4$ (таблиця 2.6, таблиця 2.8). Вони вираховуються з трикутника (чотирикутника) постійної деформації (рис. 2.3, рис. 2.4) та координат його вершин (таблиця 2.5, таблиця 2.7).

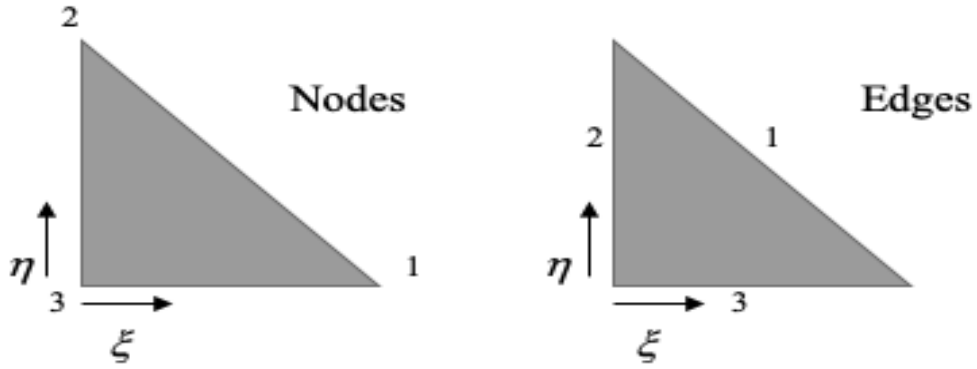


Рисунок 2.3 Трикутник постійної деформації

Джерело: складено автором

Node Number	Coordinates	
	ξ	η
1	1	0
2	0	1
3	0	0

Таблиця 2.5 Координати вузлів трикутника

$N^1 = \xi$	$N^1_{,\xi} = 1$	$N^1_{,\eta} = 0$
$N^2 = \eta$	$N^2_{,\xi} = 0$	$N^2_{,\eta} = 1$
$N^3 = (1 - \xi - \eta)$	$N^3_{,\xi} = -1$	$N^3_{,\eta}(\xi, \eta) = -1$

Таблиця 2.6 Значення функцій форм трикутника постійної деформації

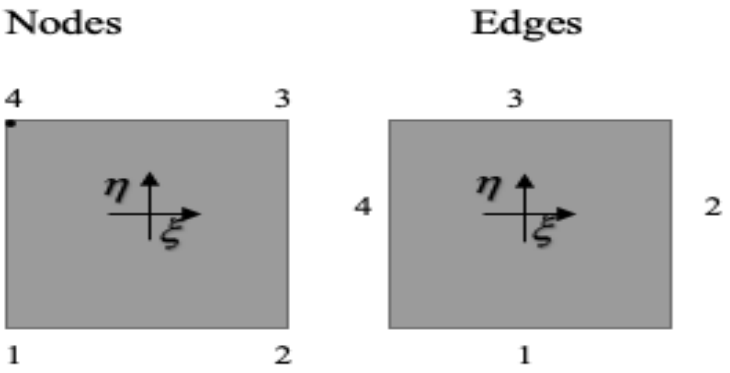


Рисунок 2.4 Чотирикутник постійної деформації

Джерело: складено автором

Node number	Coordinates	
	ξ	η
1	-1	-1
2	1	-1
3	1	1
4	-1	1

Таблиця 2.7 Координати вузлів чотирикутника

$N^1 = \frac{1}{4}(1-\xi)(1-\eta)$	$N^1_{,\xi} = -\frac{1}{4}(1-\eta)$	$N^1_{,\eta} = -\frac{1}{4}(1-\xi)$
$N^2 = \frac{1}{4}(1+\xi)(1-\eta)$	$N^2_{,\xi} = +\frac{1}{4}(1-\eta)$	$N^2_{,\eta} = -\frac{1}{4}(1+\xi)$
$N^3 = \frac{1}{4}(1+\xi)(1+\eta)$	$N^3_{,\xi} = +\frac{1}{4}(1+\eta)$	$N^3_{,\eta} = +\frac{1}{4}(1+\xi)$
$N^4 = \frac{1}{4}(1-\xi)(1+\eta)$	$N^4_{,\xi} = -\frac{1}{4}(1+\eta)$	$N^4_{,\eta} = +\frac{1}{4}(1-\xi)$

Таблиця 2.8 Значення функцій форм чотирикутника постійної деформації

Підставивши значення $N^1_\xi, \dots, N^4_\eta$, отримаємо Якобіан, який далі буде використовуватись для генерації матриці жорсткості вузла.

Тепер можемо записати її рівняння:

$$K^{ij}_{ac} = \sum_{gp=1}^{GPE} \left[J^T * N^i_\xi\right]_b E_{abcd} \left[J^T * N^j_\eta\right]_d * \det(J) * a_{gp},$$

де $a, b, c, d = \overline{0, PD}$ напрямки в елементі.

Це рівняння дійсне для чотирикутників, а для трикутників потрібно результат розділити на 2. В рівнянні відомо все, окрім E_{abcd}, тож розгляньмо визначення даного компонента.

$$E_{abcd} = \frac{E}{2 * (1+\nu)} * (\delta_{ad} * \delta_{bc} + \delta_{ac} + \delta_{bd}) + \frac{E\nu}{1-\nu^2} * \delta_{ab} * \delta_{cd},$$

де $\delta_{ii} = \begin{cases} 0, if\ i == j \\ 1, if\ i\ != j \end{cases}$, а E та ν задаються користувачем.

Наприклад, значення E_{abcd} при $PD = 2$:

$$E_{1111} = \frac{E}{1-\nu^2} \qquad E_{1112} = 0 \qquad E_{1121} = 0 \qquad E_{1122} = \frac{E}{1-\nu^2}$$

$$E_{1211} = 0 \qquad E_{1212} = \frac{E}{2*(1+\nu)} \qquad E_{1221} = \frac{E}{2*(1+\nu)} \qquad E_{1222} = 0$$

$$E_{2111} = 0 \qquad E_{2112} = \frac{E}{2(1+\nu)} \qquad E_{2121} = \frac{E}{\nu*(1+\nu)} \qquad E_{2122} = 0$$

$$E_{2211} = \frac{E\nu}{1-\nu^2} \qquad E_{2212} = 0 \qquad E_{2221} = 0 \qquad E_{2222} = \frac{E}{1-\nu^2}$$

Тоді маємо $K^{ij}_{[2x2]} = \begin{bmatrix} K^{ij}_{xx} & K^{ij}_{xy} \\ K^{ij}_{yx} & K^{ij}_{yy} \end{bmatrix}$ – матриця жорсткості вузла.

Матриця жорсткості елемента для PD = 2 та NpE = 3 має наступний вигляд:

$$K = \begin{bmatrix} K^{11}K^{12}K^{13}K^{14} \\ K^{21}K^{22}K^{23}K^{24} \\ K^{31}K^{32}K^{33}K^{34} \\ K^{41}K^{42}K^{43}K^{44} \end{bmatrix}.$$

Підставимо відповідні значення матриць вузлів та отримаємо повну матрицю жорсткості елемента розміром 6x6.

2.4 Глобальна матриця жорсткості та застосування додаткових умов

Перш ніж продовжити, пропоную розглянути види деформацій тіла, які будуть підтримуватись калькулятором.

Це розтяг (рис. 2.5), розширення(рис. 2.6) та зсув (рис. 2.7). Встановимо значення прикладеної сили рівним 0.1 N (Ньютон).

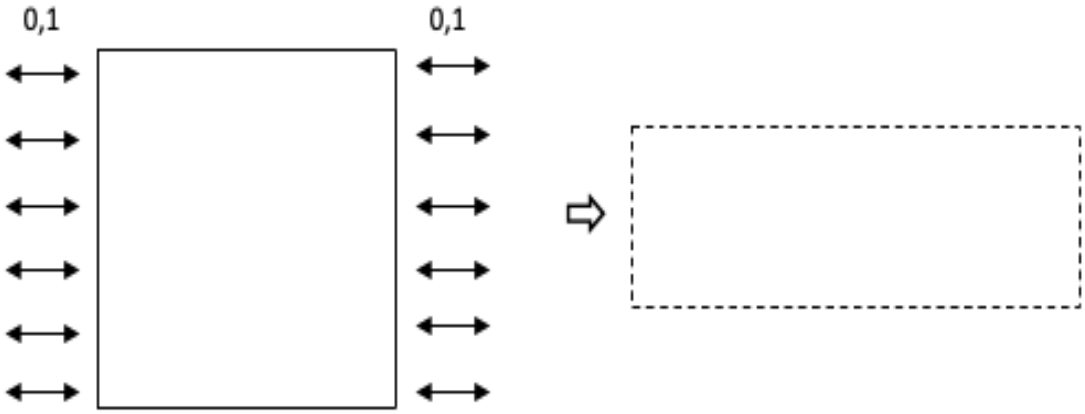


Рисунок 2.5 Схема деформації для розтягу

Джерело: складено автором

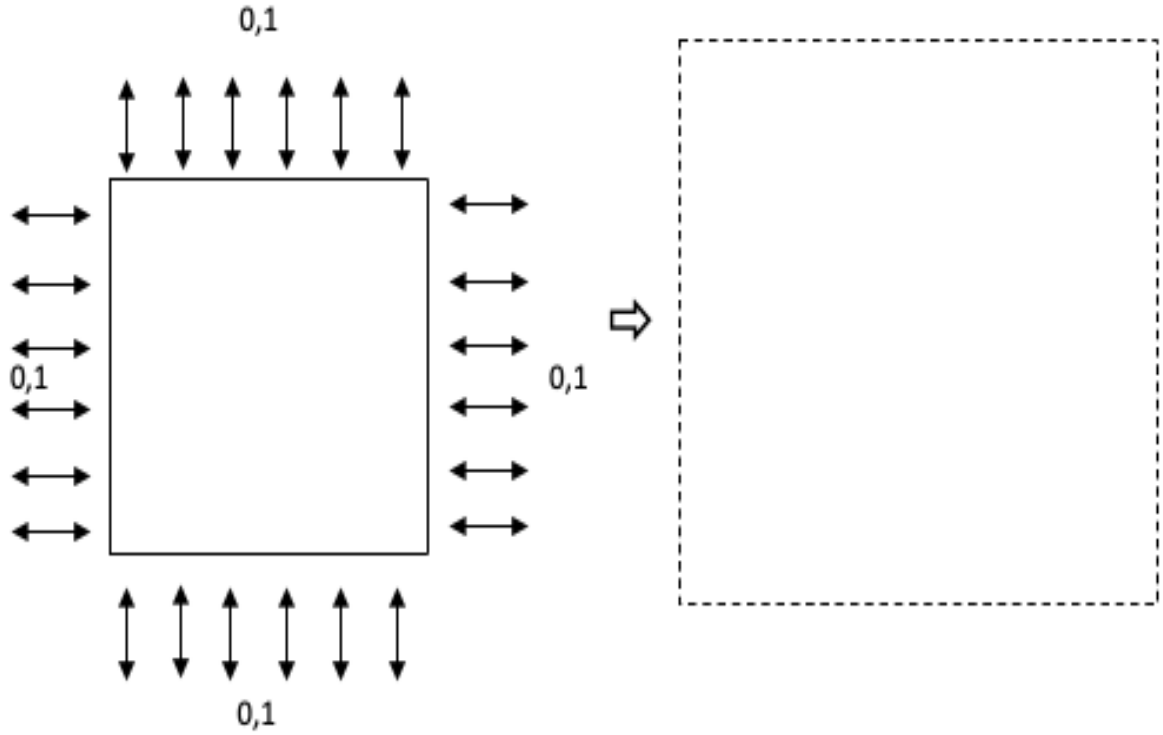


Рисунок 2.6 Схема деформації для розширення

Джерело: складено автором

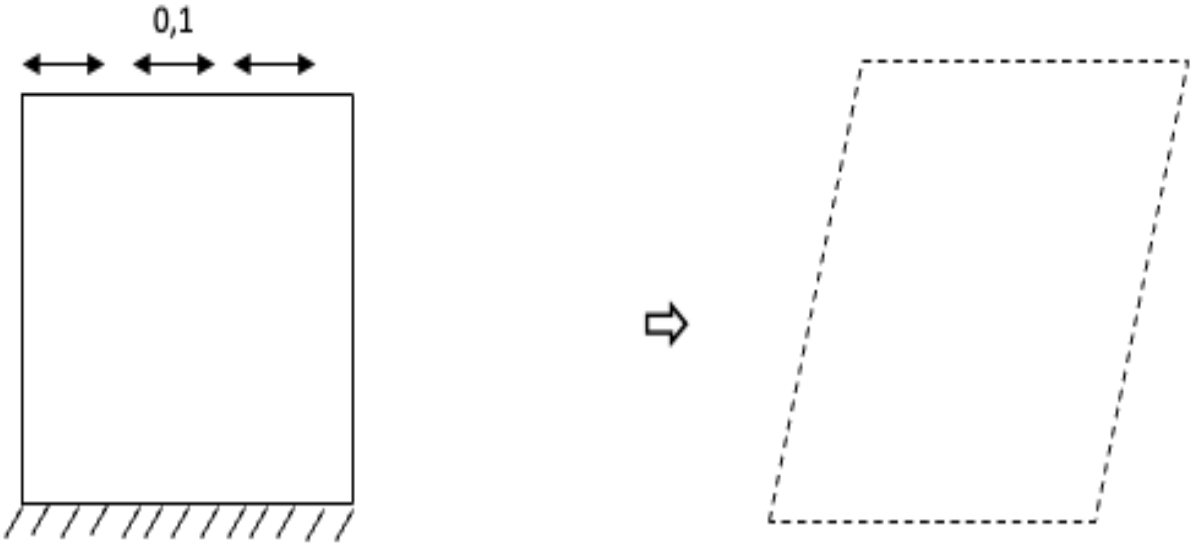


Рисунок 2.7 Схема деформації для зсуву

Джерело: складено автором

Для кожного типу будуємо власні граничні умови:

- Для розтягу:

- Вузли, які знаходяться на лівому та правому ребрах, будуть мати граничні умови Діріхле: $(-1, -1)$, а вектори прикладених сил будуть дорівнювати $(-0.1, 0)$ та $(0.1, 0)$ відповідно.

- При розширенні всі вузли на ребрах об'єкта матимуть граничні умови Діріхле: $(-1, -1)$, вектор сил: $0.1 * (node.x, node.y)$.

- Для зсуву:

- Вузли, які знаходяться на основі мають граничні умови Діріхле: $(-1, -1)$, а вектор сил: $(0,0)$.

- Вузли, які знаходять на верхньому ребрі будуть мати граничні умови Діріхле: $(-1, -1)$ та вектор прикладених сил: $(0.1, 0)$.

Всі решта вузлів для кожного з типів деформації матимуть граничні умови: (1,1), вектор прикладених сил: (0, 0).

Проведемо збірку глобальної матриці жорсткості $K_{[NoN*PD\ x\ NoN*PD]} = \begin{bmatrix} \cdots & & \\ \vdots & \ddots & \vdots \\ & \cdots & \end{bmatrix}$, використовуючи раніше знайдені матриці жорсткості елементів.

Тепер необхідно створити масиви прикладених сил та зсувів на основі даних, які ми отримали на попередніх кроках даного етапу.

В результаті маємо все необхідне для створення СЛАР та її розв’язку.

2.5 Розробка інтерфейсу

Оскільки єдиним методом взаємодії користувача з програмою буде інтерфейс, він повинен бути інтуїтивно зрозумілим, однозначним, зручним та забезпечувати доступ до всіх необхідних інструментів серверної частини додатка. На цьому етапі потрібно поставити себе на місце користувача та крок за кроком виконати уявне завдання, щоб краще зрозуміти послідовність використання функціонала аплікації.

Розіб’ємо процес використання калькулятора на три етапи:

- перший – ініціалізація задачі, введення вхідних даних, створення та модифікація сітки скінченних елементів;
- другий – встановленні властивостей матеріалу, виборі виду трансформації;
- третій – перегляд результатів.

Для виконання першого кроку необхідно бачити сам об’єкт, тип елементів, на які його буде розбито, мати можливість вказати розмірність бажаної сітки, та, при бажанні, додати або видалити вузол, а також кнопку

для генерації сітки елементів. Створимо прототип першого блоку інтерфейсу (рис. 2.8).

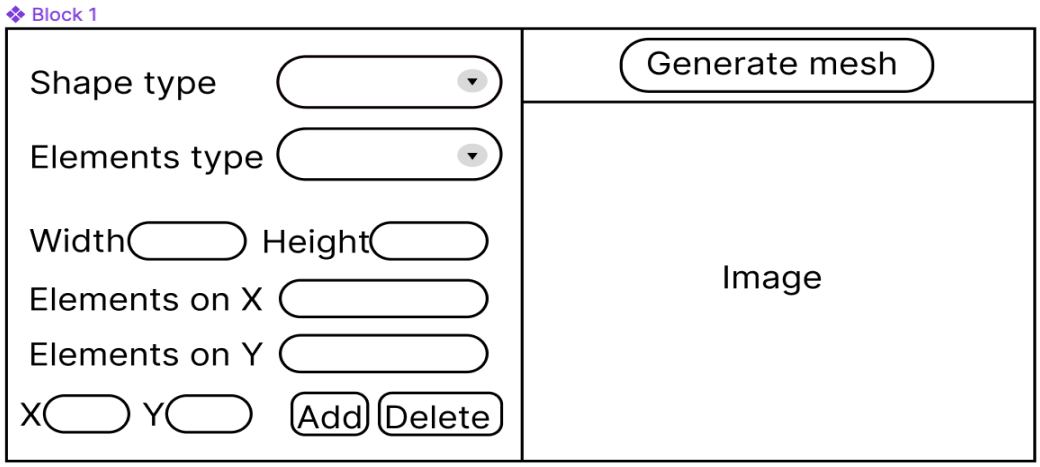


Рисунок 2.8 Прототип першого блоку користувацького інтерфейсу

Джерело: складено автором

Другий крок потребує полів для введення додаткових даних, способів вибору деформації та кнопку запуску алгоритму, також додамо кнопку для отримання текстового варіанту результатів (рис. 2.9).

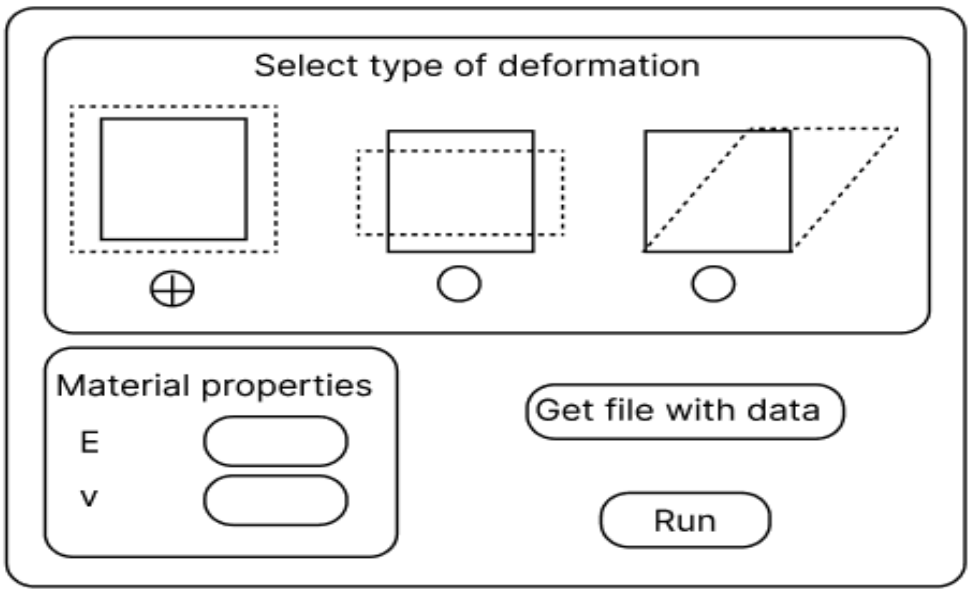


Рисунок 2.9 Прототип другого блоку користувацького інтерфейсу

Джерело: складено автором

Для третього кроку в інтерфейсі буде додатковий блок для виведення зображення із застосованим градієнтом.

2.6 Взаємодія сервера та клієнта

При розробці веб додатків необхідно однозначно визначити способи обміну інформації між клієнтом та сервером, а також сформувати структури даних, якими вони будуть оперувати. Для розробки розробки серверної частини калькулятора я буду використовувати Python фреймворк Flask. Обмін даних буде відбуватись за протоколом http, формат даних для передачі – JSON.

Серверна частина розроблена з оглядом на стандарти REST архітектури. При розробці програми методу скінченних елементів також застосовувалась методологія ООП та принципів SOLID.

2.7 Перевірка програми на тестовій задачі

Щоб переконатись в коректності роботи імплементації методу, було вирішено випробувати її на тестовій задачі, яка складається з наступних даних:

1) розмірність об’єкту 1×1;
2) обидва значення elements_on_X та elements_on_Y дорівнюють 10;
3) тип скінченних елементів – чотирикутник;
4) вид деформації – розтяг;
5) властивості матеріалу: v (коефіцієнт Пуассона) дорівнює 0,33, а E (модуль Юнга) становить 3;

6) також додамо новий параметр R дорівнює 0,3.

Параметр R визначає радіус внутрішнього порожнього кола, що знаходиться в середині об’єкту.

Проміжними результатами будуть сітка розбиття елементів, матриця жорсткості першого елемента, глобальна матриця жорсткості та матриця зсувів та напруги елементів.

Основними результатами будуть два графіки, які візуалізують матрицю зсувів по осі X та напруги елементів.

Тестування буде виконуватись в Spyder IDE та без використання клієнтської частини додатку задля зручності, швидкості та кращого доступу до даних. Сітка розбиття вказана на рис. 2.10.

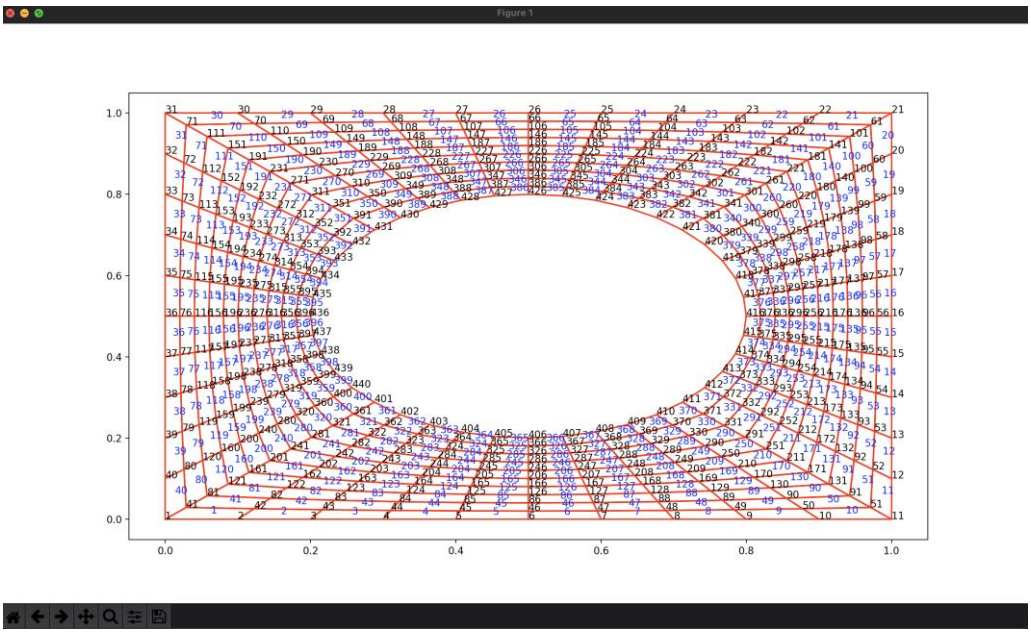
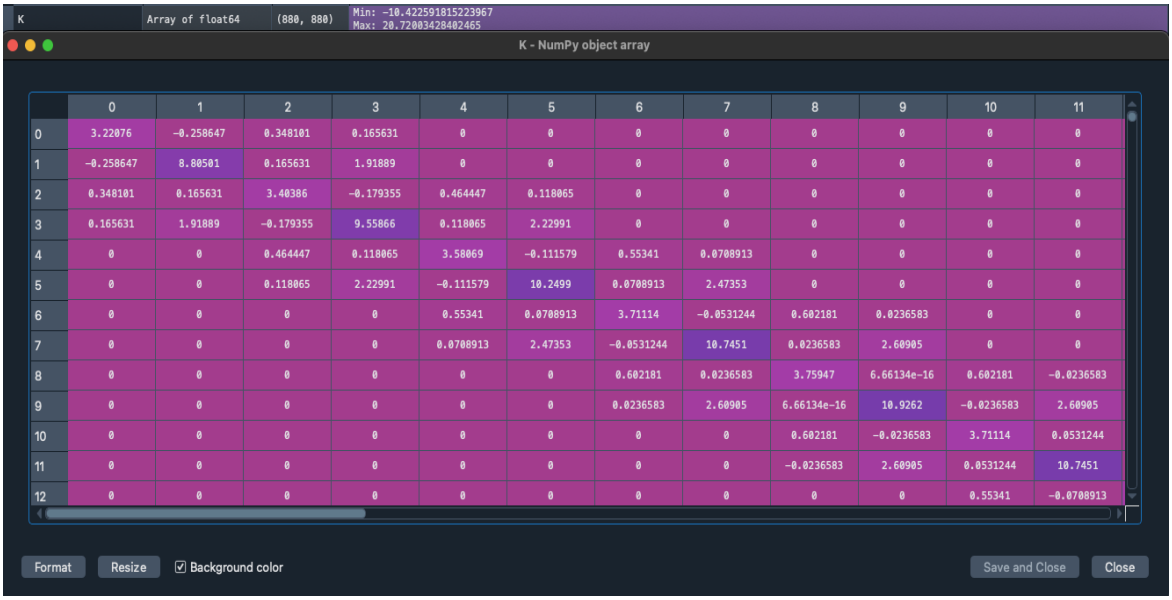


Рисунок 2.10 Сітка розбиття фізичного об’єкта

Проаналізувавши рисунки 2.11, 2.12, 2.13, 2.14, 2.15 можемо пересвідчитись у коректності розмірності матриць та, при бажанні, вручну порахувати значення щоб бути впевненим у правильності обчислень.

	0	1	2	3	4	5	6	7
0	1.05661	0.397859	0.216355	0.214436	−0.339492	−0.468852	−0.933471	−0.143442
1	0.397859	2.50408	0.214436	1.57793	−0.468852	−0.584295	−0.143442	−3.49771
2	0.216355	0.214436	2.01758	−0.702892	−1.02201	−0.134677	−1.21192	0.623133
3	0.214436	1.57793	−0.702892	5.72625	−0.134677	−3.73475	0.623133	−3.56943
4	−0.339492	−0.468852	−1.02201	−0.134677	1.17319	0.386317	0.188318	0.217212
5	−0.468852	−0.584295	−0.134677	−3.73475	0.386317	2.81618	0.217212	1.50287
6	−0.933471	−0.143442	−1.21192	0.623133	0.188318	0.217212	1.95707	−0.696902
7	−0.143442	−3.49771	0.623133	−3.56943	0.217212	1.50287	−0.696902	5.56427

Рисунок 2.11 Матриця жорсткості першого елемента



	0	1	2	3	4	5	6	7	8	9	10	11
0	3.22076	−0.258647	0.348101	0.165631	0	0	0	0	0	0	0	0
1	−0.258647	8.80501	0.165631	1.91889	0	0	0	0	0	0	0	0
2	0.348101	0.165631	3.40386	−0.179355	0.464447	0.118065	0	0	0	0	0	0
3	0.165631	1.91889	−0.179355	9.55866	0.118065	2.22991	0	0	0	0	0	0
4	0	0	0.464447	0.118065	3.58069	−0.111579	0.55341	0.0708913	0	0	0	0
5	0	0	0.118065	2.22991	−0.111579	10.2499	0.0708913	2.47353	0	0	0	0
6	0	0	0	0	0.55341	0.0708913	3.71114	−0.0531244	0.602181	0.0236583	0	0
7	0	0	0	0	0.0708913	2.47353	−0.0531244	10.7451	0.0236583	2.60905	0	0
8	0	0	0	0	0	0	0.602181	0.0236583	3.75947	6.66134e-16	0.602181	−0.0236583
9	0	0	0	0	0	0	0.0236583	2.60905	6.66134e-16	10.9262	−0.0236583	2.60905
10	0	0	0	0	0	0	0	0	0.602181	−0.0236583	3.71114	0.0531244
11	0	0	0	0	0	0	0	0	−0.0236583	2.60905	0.0531244	10.7451
12	0	0	0	0	0	0	0	0	0	0	0.55341	−0.0708913

Рисунок 2.12 Глобальна матриця жорсткості

K	Array of float64	(880, 880)	Min: -10.422591815223967 Max: 20.72003428402465

Рисунок 2.13 Параметри глобальної матриці жорсткості

stress_xx | Array of float64 | (4, 400) | Min: -0.11136449900632536 Max: 1.2871965520387056

stress_xx - NumPy object array

	0	1	2	3	4	5
0	0.798703	0.666705	0.606481	0.398521	0.163946	0.15995
1	0.776706	0.701615	0.634067	0.404584	0.15995	0.163946
2	0.719802	0.670686	0.596354	0.413944	0.227141	0.230527
3	0.745917	0.636756	0.569809	0.40741	0.230527	0.227141

Рисунок 2.14 Матриця напруги вузлів елементів

disp_x | Array of float64 | (4, 400) | Min: -0.10253924612881651 Max: 0.102539246128816

disp_x - NumPy object array

	0	1	2	3	4	5	6
0	-0.1	-0.0702617	-0.0443668	-0.020629	-0.00541123	-2.381e-16	0.00541123
1	-0.0702617	-0.0443668	-0.020629	-0.00541123	-2.381e-16	0.00541123	0.020629
2	-0.0684499	-0.0449557	-0.0239732	-0.00881164	-1.81068e-16	0.00881164	0.0239732
3	-0.0933008	-0.0684499	-0.0449557	-0.0239732	-0.00881164	-1.81068e-16	0.00881164

Рисунок 2.15 Матриця зсувів вузлів елементів

Розглянувши всі проміжні обчислення та переконавшись що вони коректні, можемо перейти до перегляду фінальних результатів (рис. 2.16, рис. 2.17).

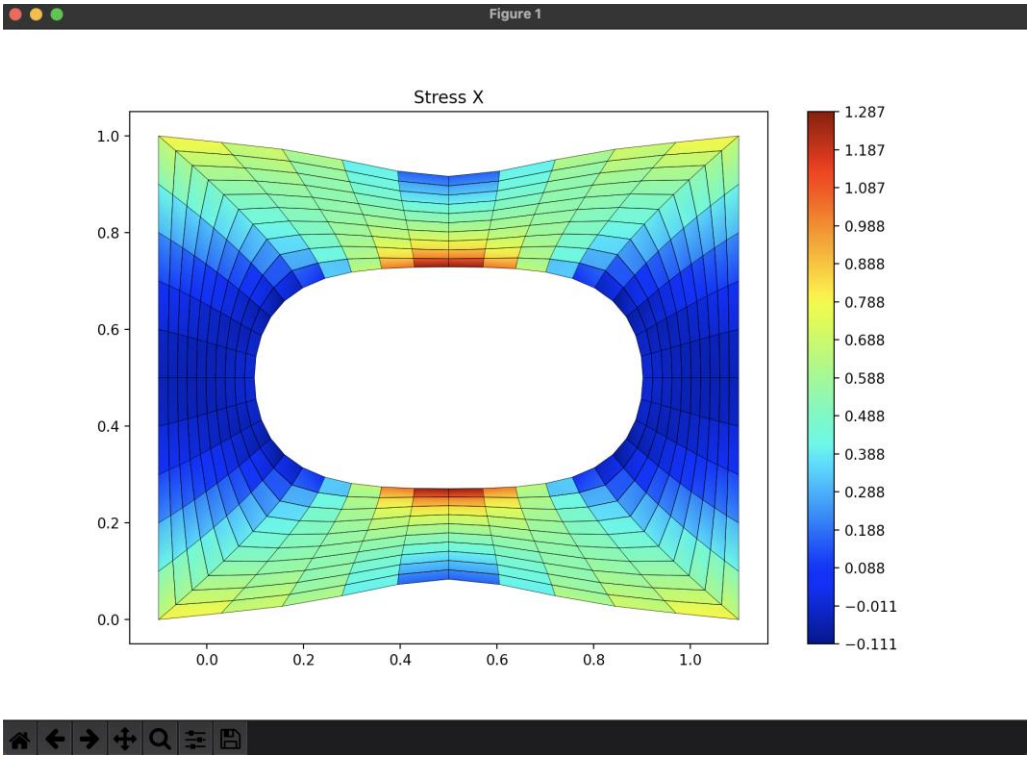


Рисунок 2.16 Графік напруг елементів

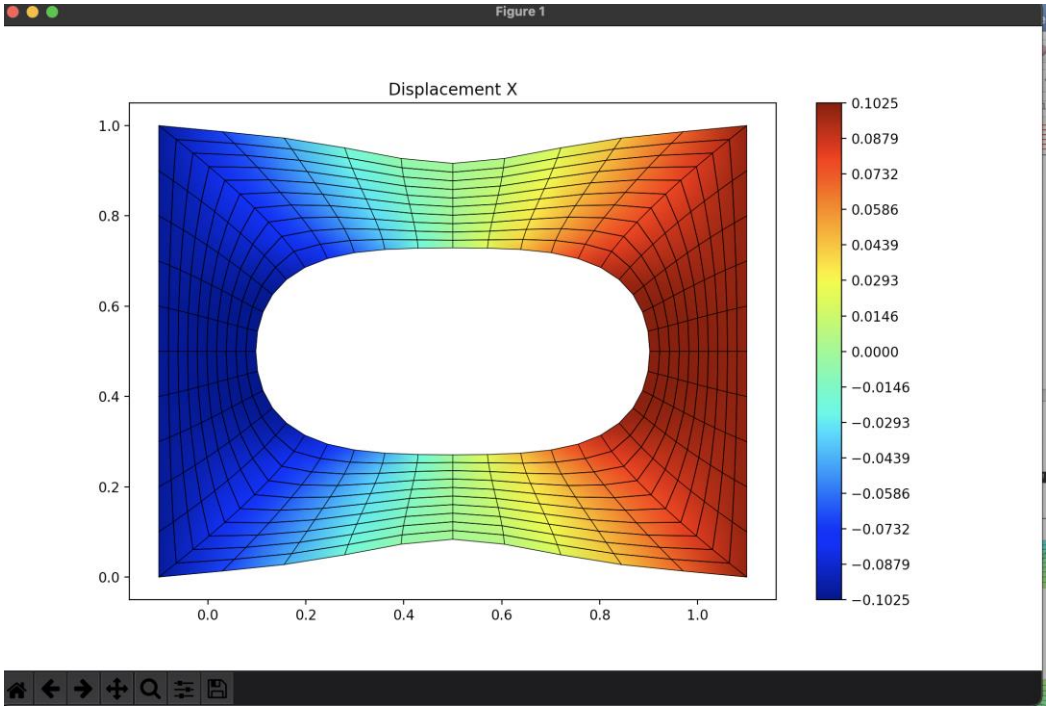


Рисунок 2.17 Графік зсувів елементів

2.8 Тестування веб додатка

Останнім етапом розробки будь-якого застосунка є процес тестування усього доступного функціоналу. Для цього перейдемо за адресою http://127.0.0.1:5000/mash, за якою знаходиться сторінка введення вхідних даних для генерації сітки (рис. 2.18).

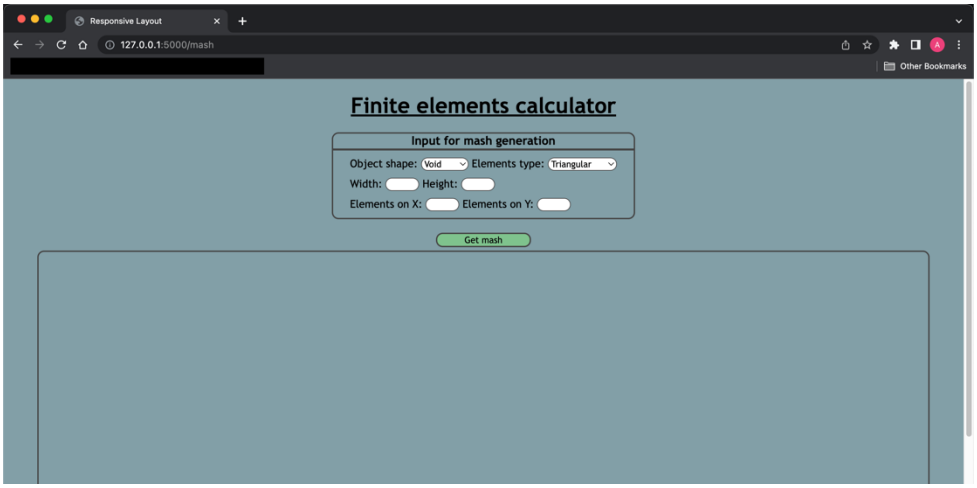


Рисунок 2.18 Сторінка введення даних для генерації сітки скінченних елементів

Також переконаймось у можливості вибору форми об’єкту та тиру елементів (рис. 2.19, рис 2.20).

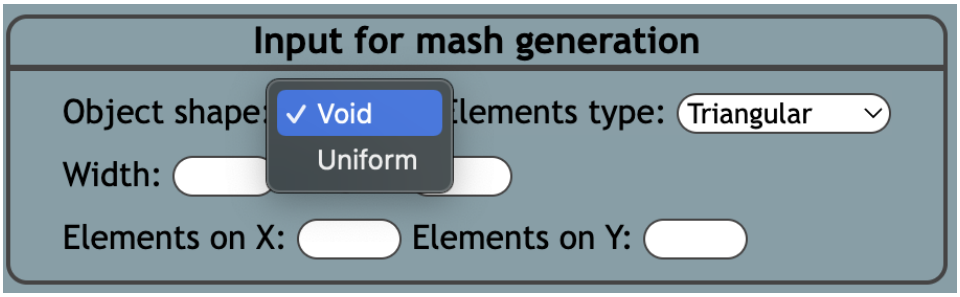


Рисунок 2.19 Вибір форми об’єкта

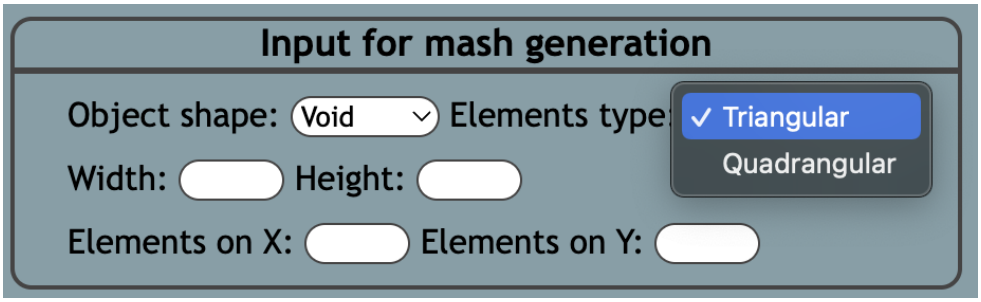


Рисунок 2.20 Вибір типу елементів

Введемо дані та перевіримо генерацію сітки елементів натиснувши зелену кнопку ‘Get mash’(рис 2.21).

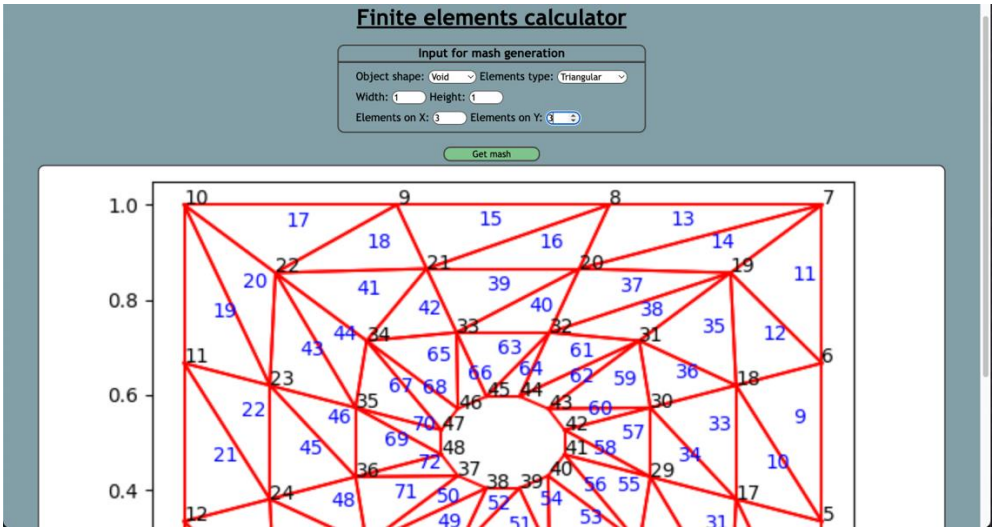


Рисунок 2.21 Результат генерації сітки

Перейдемо на адресу http://127.0.0.1:5000/fem_processor та побачимо сторінку введення даних для методу скінченних елементів (рис. 2.22).

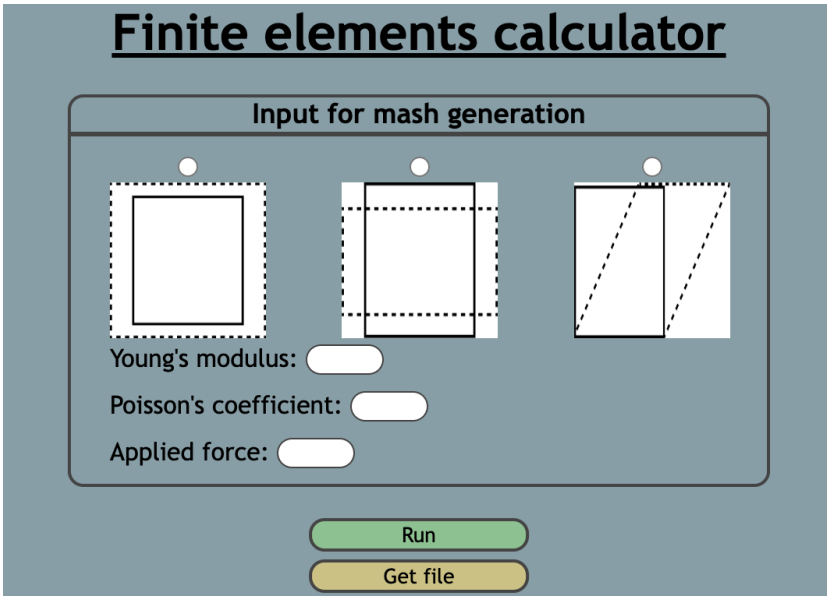


Рисунок 2.22 Інтерфейс введення даних для МСЕ

Введемо дані та натиснемо зелену кнопку ‘Run’ в результаті отримаємо кольоровий графік зміщень елементів (рис. 2.23, рис. 2.24).

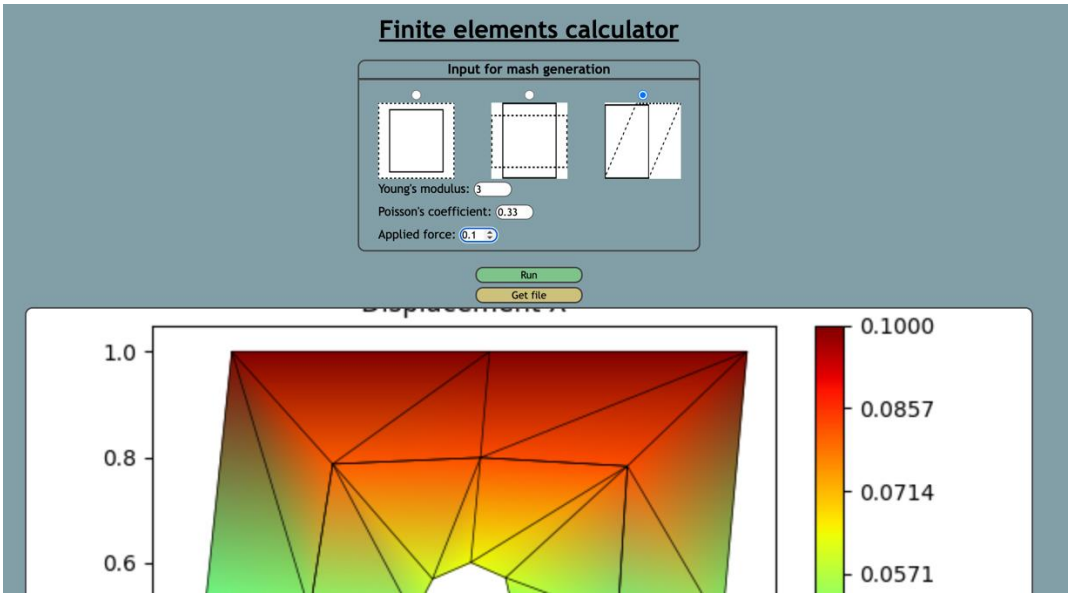


Рис 2.23 Результат виконання методу

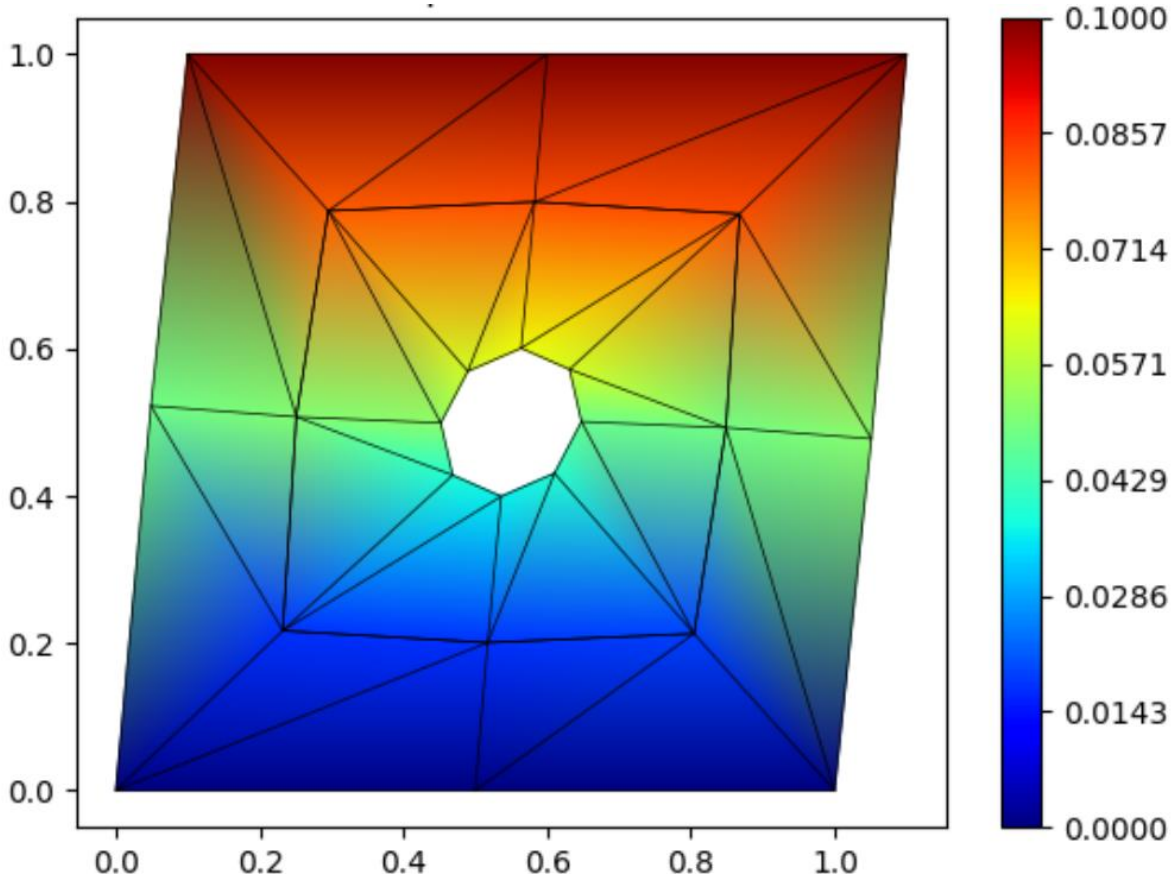


Рисунок 2.24 Повна матриця зсувів елементів

ВИСНОВКИ

Раціональність застосування методу скінченних елементів у сучасних задачах обчислюваної механіки обумовлена необхідністю вирішувати комплексні завдання, які потребують чисельного обчислення. Завдяки своєму всебічному підходу до вирішення поставлених питань, метод скінченних елементів є корисним у багатьох наукових напрямах, наприклад, прикладній фізиці, біомеханіці, моделюванні тощо.

Метод скінченних елементів є більш доцільним для промислового призначення, аніж для особистого. При вирішенні комплексних задач комп'ютерного моделювання, досліджуваний метод дає змогу швидко та якісно, з точки зору точності, отримати потрібний результат.

Проте більшість програмного забезпечення для використання методу скінченних елементів має надлишкову функціональність та високий поріг входження, що являється досить складним для особистого застосування.

Задля опрацювання нового, зручного для вирішення індивідуальних потреб засобу використання алгоритму, мною було розроблено онлайн калькулятор. Дана праця та створений додаток можуть слугувати інструкцією для розробки аналогічних проектів.

У ході роботи мною було проаналізовано принцип роботи та основні прийоми методу скінченних елементів та використання його для програмної реалізації, розглянуто особливості застосування методу, розроблено та реалізовано веб-додаток для обчислення двовимірних задач, виконані тестові розрахунки за допомогою розробленого доданку.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Д.В. Іванов, А.В. Доль. Введення в метод скінченних елементів: навчально – методичний посібник для студентів природничо-наукових дисциплін. – Саратов: Аміріт, 2016.

2. Ю. А. Сагдєєва, С.П. Кописов, А.К.Новіков. Введення в метод скінченних елементів: методичний посібник. Іжевськ: вид-во «Удмуртський університет», 2011.

3. В.Б. Андрєєв. Лекції по методу скінченних елементів: навчальний посібник. - М: Видавничий відділ факультету ОМіК МГУ ім. М.В. Ломоносова; МАКС Прес, 2010. – 2-е видання.

4. Г. Стренг, Дж. Фікс. Теорія методу скінченних елементів. – М: Вид-во «Мир», 1977.

5. Silke Prohl. Finite Element Methods for Partial Differential Equations for Option Pricing. – 2019.

6. Р.З. Даутов, М.М. Карчевський. Введення в теорію метода скінченних елементів: навчальний посібник. – Казань: Казанський державний університет ім. В.І. Улянова-Леніна, 2004.

7. Finite elements analysis and implementation. Ел. ресурс. Режим доступу: https://finite-element.github.io/index.html.