МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра програмної інженерії

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему: **«РОЗРОБКА АВТОМАТИЗОВАНОЇ СИСТЕМИ СКІНЧЕННО-ЕЛЕМЕНТНОГО АНАЛІЗУ НА БАЗІ PYTHON»**

Виконала: студентка 2 курсу, групи 8.1211-з
спеціальності 121 інженерія програмного забезпечення
(шифр і назва спеціальності)
освітньої програми інженерія програмного забезпечення
(назва освітньої програми)
М.В. Маляревич
(ініціали та прізвище)

Керівник завідувач кафедри фундаментальної та прикладної математики, професор, д.т.н. Гребенюк С.М.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Рецензент доцент кафедри комп'ютерних наук, доцент, к.т.н. Решевська К.С.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя – 2022

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра програмної інженерії

Рівень вищої освіти магістр

Спеціальність 121 інженерія програмного забезпечення
(шифр і назва)

Освітня програма інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри програмної інженерії, к.ф.-м.н., доцент

_____ Лісняк А.О.
(підпис)

“ ____ ” _____ 2022 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТЦІ

Маляревич Маргариті Віталіївні
(прізвище, ім’я та по-батькові)

1. Тема роботи Розробка автоматизованої системи скінченно-елементного аналізу на базі Python

керівник роботи Гребенюк Сергій Миколайович, д.т.н., професор
(прізвище, ім’я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 10 » травня 2022 року № 514-с

2. Строк подання студентом роботи

3. Вихідні дані до роботи 1. Постановка задачі.
2. Перелік літератури.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Постановка задачі.
2. Основні теоретичні відомості.
3. Автоматизація системи скінчено-елементного аналізу.

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)
презентація

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання _____

К А Л Е Н Д А Р Н И Й П Л А Н

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.	22.05.2022	
2.	Збір вихідних даних.	30.06.2022	
3.	Обробка методичних та теоретичних	25.09.2022	
	джерел.		
4.	Розробка першого та другого розділу.	16.10.2022	
5.	Розробка третього розділу.	20.11.2022	
6.	Оформлення та нормоконтроль	27.11.2022	
	кваліфікаційної роботи магістра.		
7.	Захист кваліфікаційної роботи.	16.12.2022	

Студент _____ М.В. Маляревич
(підпис) (ініціали та прізвище)

Керівник роботи _____ С.М. Гребенюк
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ А.В. Столярова
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота магістра «Розробка автоматизованої системи скінченно-елементного аналізу на базі Python»: 42 с., 10 рис., 2 табл., 12 джерел, 1 додаток.

МАТРИЦЯ ЖОРСТКОСТІ, МЕТОД СКІНЧЕНИХ ЕЛЕМЕНТІВ, ПРОГРАМНИЙ КОД, САПР, NUMPY, PANDAS, PYTHON.

Об’єкт дослідження – автоматизована система скінченно-елементного аналізу на базі Python.

Мета роботи: створення САПР для визначення напружено-деформованого стану двовимірних елементів конструкцій на базі Python.

Метод дослідження – аналітичний.

SUMMARY

Master’s qualifying paper «Development of the Finite Element Analysis Automated System using Python»: 42 pages, 10 figures, 2 tables, 12 references, 1 supplement.

STIFFNESS MATRIX, FINITE ELEMENT METHOD, SOFTWARE CODE, CAD, NUMPY, PANDAS, PYTHON.

Object of the study – is an automated finite element analysis system based on Python.

Aim of the study: creation of CAD for determining the stress-strain state of two-dimensional structural elements based on Python.

Method of research – analytical.

ЗМІСТ

ВСТУП

Метод скінчено-елементного аналізу – це чисельний метод для розв’язання інженерних та математичних задач побудованих на диференціальних рівняннях. Спочатку цей метод використовувався для розв’язання диференційних рівнянь для визначення напружень, але на сьогоднішній день він широко використовується в таких напрямках як будівництво, аерокосмічна діяльність, телекомунікації, біомеханіка та багато інших.

Реалізація методу скінчено-елементного аналізу лежить саме на основі програмної розробки. Наприкінці 1960-х років для Національного управління з аеронавтики і дослідження космічного простору (National Aeronautics and Space Administration (NASA)) розробили програмне забезпечення для аналізу методу скінчених елементів NASTRAN (NASA STRuctrual Analysis), яке було реалізоване на мові програмування Fortran.

На сьогоднішній день метод скінчено-елементного аналізу широко використовується. Завдяки швидкому розвитку світу програмного забезпечення цей метод можна реалізувати на таких мовах програмування як C++, .NET, Python та інші.

В роботі буде показана розробка автоматизованої системи скінчено-елементного аналізу на базі Python.

Python є об’єктно-орієнтовною мовою програмування і був розроблений в 1990 році. З’явившись порівняно пізно, Python створювався під впливом багатьох мов програмування:

– Fortran;
– Java;
– C, C++;
– TypeScript.

Саме тому Python містить в собі все необхідне для реалізації методу скінчено-елементного аналізу. Це можна реалізувати за допомогою таких

бібліотек як math, numpy, GetFEM. Саме ці бібліотеки створені для роботи з математичними задачами та методом скінчених елементів.

1 МЕТОД СКІНЧЕНО-ЕЛЕМЕНТНОГО АНАЛІЗУ ТА ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

1.1 Метод скінчено-елементного аналізу

Метод скінченних елементів є ефективним чисельним методом розв’язання інженерних і фізичних задач. Область його застосування тягнеться від аналізу напружень в конструкціях літаків або автомобілів до розрахунку таких складних систем, як атомна електростанція. З його допомогою розглядається рух рідини по трубах, через дамби, в пористих середовищах, досліджується перебіг газу, що стискається, вирішуються завдання електростатики, аналізуються коливання систем. Метод скінченних елементів є чисельним методом розв’язання диференціальних рівнянь, що зустрічаються у фізиці й техніці. Виникнення цього методу пов’язане з розв’язанням космічних задач. Область застосування методу скінченних елементів істотно поширилася, коли було показано, що рівняння, які описують елементи у задачах будівельної механіки, розповсюдження тепла і гідромеханіки аналогічні. Метод скінченних елементів з чисельної процедури розв’язання задач будівельної механіки перетворився на загальний метод чисельного розв’язання диференціальних рівнянь. Основна ідея цього методу полягає в тому, що будь-яку безперервну величину, таку як температура, тиск і переміщення, можна апроксимувати дискретною моделлю, яка будується на безлічі кусково-безперервних функцій. У загальному випадку безперервна величина наперед не відома, і потрібно визначити значення цієї величини в деяких внутрішніх точках області. Дискретну модель дуже легко побудувати, якщо спочатку припустити, що числові значення цієї величини в кожній внутрішній області відомі. Після цього можна перейти до загального випадку. Отже, при побудові дискретної моделі безперервної величини поступають таким чином [1]:

– у даній області фіксується скінченна кількість точок: ці точки

називаються вузловими точками або вузлами;

– значення безперервної величини в кожній точці вважається невідомим, яке має бути визначено;

– область визначення безперервної величини розбивається на скінченну кількість областей, які називаються елементами: ці елементи мають спільні вузлові точки і в сукупності апроксимують форму області;

– безперервна величина апроксимується на кожному елементі поліномом або іншою функцією, які визначається за допомогою вузлових значень цієї величини.

Для кожного елемента визначається свій поліном, але поліноми підбираються так, щоб зберіглася безперервність величини уздовж меж елемента (його називають апроксимуючою функцією елемента). Вибір форми елементів і їх апроксимуючих функцій для конкретних задач визначає точність наближеного розв’язку і залежить від винахідливості і майстерності інженера [2].

1.2 Fortran та Python

Python і Fortran є мовами програмування високого рівня. Звичайно, Fortran був першою мовою програмування високого рівня, і її винайшов Джон Бекус для IBM (International Business Machines) у 1954 році [3]. Python, розроблений Гвідо ван Россумом і вперше випущений у 1991 році, є мовою програмування загального призначення, що означає, що вона не спеціалізується на будь-якому конкретному типі проблеми. Python зазвичай використовується для аналізу даних, штучного інтелекту та машинного навчання, створення веб-сайтів або для візуалізації даних тощо. Однак безпосередньо мова програмування разом зі стандартною бібліотекою не підготовлена для чисельних обчислень. Отже, для розширення його можливостей використовуються переважно чотири бібліотеки: numpy, scipy, matplotlib і pandas. Fortran, що означає Formula Translation (переклад формул), був спеціально створений для наукових обчислень.

Упродовж послідовних стандартів, розроблених за час його історії, усі функції, необхідні для виконання наукових обчислень, вбудовані. Із загальної точки зору, Fortran є статично скомпільованою мовою, суворо типізованою, нативно паралельною, і вона була спроєктована для високопродуктивних обчислень. Python, навпаки, є інтерпретованою, динамічно типізованою та слабо типізованою мовою.

Python і Fortran дозволяють розробляти величезні програми на основі різних модулів. У Python модулі – це вихідні файли Python, що містять функції, які можуть бути імпортовані або легко використовуватись з іншого файлу Python. Його можна вважати Python-бібліотекою. У Fortran модулі задаються назвою «module xxx» і можуть зберігатися у вихідних файлах Fortran. Щоб дотримуватися тієї ж методології, що й Python, бажано назвати вихідний файл Fortran такою ж назвою, що й модуль, зберігаючи лише один модуль на файл. Цей файл Fortran містить функції. Оскільки Python є нетипізованою мовою, інтерфейс функцій невідомий під час імпорту. Це палиця з двома кінцями. Сприятливим наслідком є та сама функція, що використовується для різних типів даних, якщо визначено відповідні операції. Негативним наслідком є те, що помилки, пов’язані з неправильним використанням функцій, виявляються лише під час виконання. Fortran навпаки має явний інтерфейс для всіх публічних функції, які містяться в якомусь модулі, коли він використовується. Цей явний інтерфейс дозволяє виправити помилки реалізації під час компіляції. Однак потрібно перезавантаження під час виклику цих функцій із новими типами. В обох мовах модульність – це стовп у функціональному програмуванні, створюючий абстракції в ієрархічній формі.

Наприклад, щоб розв’язати нелінійну систему рівнянь у частинних похідних, можна визначити наступні рівні або модулі:

– прикладний рівень: розв’язати нелінійну систему рівнянь у частинних похідних;

– рівень дискретизації: метод дискретизації рівнянь у частинних похідних;

– нелінійний розв’язок: функції для розв’язування нелінійних систем

рівнянь, таких як Метод Ньютона;

– лінійна алгебра: методи розв'язування лінійних систем рівнянь.

Вихідна комплексна задача формується функціональним складом на більш дрібні або простіші проблеми чи функції. Одна з головних цілей модульності у функціональному програмуванні – це здатність будувати різні методи різних модулів без знання функціональності інших модулів. Крім того, потрібно докласти додаткових зусиль для визначення ієрархічної структури.

Покажчик (A pointer) – це змінна, яка зберігає адресу пам'яті іншої змінної або функції. Наприклад, речення a = 10.4 створює внутрішнє представлення числа типу float 10.4 і зберігає його в деякій частині пам’яті комп’ютера. Після чого цей номер зберігається, мітка a містить напрямок пам’яті, де зберігається 10.4. Кажуть, що змінна a вказує на число 10.4 [4]. Покажчики – це дуже потужна концепція для оптимізації пам’яті, компактного запису коду та передачі аргументів масиву до різних функцій. Однак їх важко зрозуміти і вони є дуже небезпечними на початку. Їх використання повинно бути обмежено для досвідчених користувачів. У Python все є внутрішнім покажчиком, і необхідно зрозуміти як це працює, щоб уникнути непорозумінь і дивної поведінки коду чи помилок. У мові Fortran є покажчики, але вони мають обмеження і можуть уникнути класичних проблем і зробити їх безпечнішими. Покажчики також використовуються під час виклику функції. Різні вхідні аргументи передаються за посиланням або за значенням. Якщо вони передаються за посиланням, це означає, що передається лише їх адреса або покажчик. Якщо вони передаються за значенням, це означає, що створюється копія вихідного аргументу і створюється нова локальна змінна для цієї функції. Як тільки функція залишена, змінна знищується. Покажчики також можуть зберігати адреси функцій і можуть бути використані для створення або побудови нових функцій.

Загалом, Python є досить таки повільною мовою програмування. Це пов’язано з тим, що Python є інтерпретованою мовою програмування. Але наукові розрахунки не покладаються на суто Python. Для чисельних математичних розрахунків використовуються бібліотеки. Наприклад, бібліотека

numpy, що розшифровується як Numerical Python, робить розрахунки на основі C/C++ або Fortran. При цьому саме Python вказує те, що необхідно отримати у відповідь. В свою чергу, завдяки використанню бібліотек, код стає набагато компактніше, аніж на інших програмах. Також, Python має досить таки простий та чистий синтаксис і все ж таки є зручним для розв’язання математичних задач.

1.3 NASTRAN (NASA Structrual Analysis)

NX Nastran – це комп’ютерна програма загального призначення для аналізу на основі скінченних елементів, яка дозволяє розв’язувати широкі задачі інженерного характеру. [5] NX Nastran написано переважно мовою FORTRAN і ефективно оптимізовано для роботи, забезпечуються ідентичні результати на різноманітних комп’ютерах і операційних системах. NX Nastran містить такі можливості аналізу:

- лінійна статика;
- задачі стиску-розтягнення та вигину;
- теплообмін (стаціонарний і нестаціонарний);
- перехідна характеристика;
- частотна характеристика;
- спектр відгуку та випадковий відгук;
- геометрична та фізична нелінійна статика;
- оптимізація дизайну (включаючи оптимізацію динаміки та форми);
- композиційні матеріали;
- акустична реакція;
- аеропружність;
- суперелементи;
- комплексний аналіз;
- осесиметричний аналіз;
- циклічна симетрія;

– p-елементи.

Програмне забезпечення NX Nastran складається з великої кількості побудованих блоків, які називаються модулями. Кожен модуль – це набір підпрограм FORTRAN, призначених для виконання певного завдання, наприклад обробка геометрії моделі, складання матриць, застосування обмежень, розв'язання матричних проблем і розрахунок вихідних даних. У NX Nastran модулі керуються внутрішньою мовою під назвою Direct Matrix Abstraction Program (DMAP), яка є мовою програмування високого рівня з власним компілятором і синтаксичним правилами. Оператор DMAP схожий на оператор виклику підпрограми у FORTRAN, з вхідною та вихідною інформацією.

Кожен тип аналізу, доступний у NX Nastran, називається «послідовністю розв'язків». Кожен розв'язок послідовності – це попередньо визначений набір сотень або тисяч команд DMAP. Після вибору послідовність розв'язків, її окремий набір команд DMAP надсилає інструкції модулям, які необхідні для виконання запитуваного розв'язку. Хоча NX Nastran містить низку попередньо визначених розв'язків, можна використовувати спеціальні «зміни» DMAP щоб або змінити існуючі послідовності розв'язків, або створити нові.

Щоб розв'язати скінченно-елементну модель у NX Nastran, необхідно:

– створити вхідний файл;

– використати команду Nastran, щоб надіслати вхідний файл для пакетної обробки;

– переглянути свої результати.

Щоб виконати аналіз за допомогою NX Nastran, необхідно створити вхідний файл, який містить дані моделі (інформація про скінченно-елементну модель, включаючи геометрію, елементи, матеріали та навантаження) а також дані аналізу (інформація про тип аналізу, який необхідно виконати, наприклад тип аналізу та тип даних, які бажано отримати). У NX Nastran вхідним файлом є текстовий файл ASCII, який можна створити таким способом:

– автоматичне використання препроцесора, наприклад NX Advanced

Simulation;

– вручну за допомогою текстового редактора.

Вхідний файл NX Nastran містить такі п’ять окремих розділів:

– заява Nastran;

– заява про керування файлами (FMS);

– акт виконавчого контролю;

– команда Case Control;

– масове введення даних.

Загалом вхідні файли Nastran мають розширення .dat. Однак NX Nastran намагається обробляти будь-які файли, які надсилаються, незалежно від його розширень. Інші поширені розширення вхідних файлів NX Nastran включають .bdf або .blk.

Після створення вхідного файлу, він надсилається на виконання як пакетний процес у NX Nastran. Оскільки NX Nastran не є інтерактивною програмою, після надсилання вхідного файлу не маємо жодної додаткової взаємодії з програмним забезпеченням, доки завдання не буде завершено, якщо не потрібно припинити роботу. Щоб виконати NX Nastran, зазвичай використовується системна команда Nastran (у деяких випадках системний адміністратор може призначити команді іншу назву), а потім назву вхідного файлу. Наприклад, як на рисунку 1.1.

$$NASTRAN\ MODEL1A$$

Рисунок 1.1 – Назва команди

Команда Nastran також дозволяє вказати певні ключові слова для керування виконанням завдання. Формат команди Nastran такий, як представлено на рисунку 1.2.

$$\text{nastran keyword}_1 = \text{value}_1\ \text{keyword}_2 = \text{value}_2 \ldots$$

Рисунок 1.2 – Ключові слова

Коли NX Nastran завершить аналіз, можна переглянути результати. NX Nastran автоматично створює кілька різних типів файлів, які підсумовано в таблиці 1.1.

Таблиця 1.1 – Команди NX Nastran

Тип файлу	Опис
.DBALL	Містить постійні дані для запуску бази даних для повторного аналізу.
.f04	Містить інформацію про файл бази даних і підсумок виконання модуля, наприклад час початку та завершення для кожного модуля, а також розмір файлу бази даних (якщо є).
.f06	Містить результати вашого аналізу, такі як переміщення та напруження, а також будь-які діагностичні повідомлення.
.LOG	Містить системну інформацію та будь-які повідомлення про помилки системного рівня.
.MASTER	Містить постійні дані для запуску бази даних для повторного аналізу.

Окрім цих автоматично згенерованих файлів, можна вручну подати запит на створення NX Nastran таких файлів (див. табл. 1.2).

Таблиця 1.2 створення NX Nastran

Тип файлу	Опис
.pch	Punch file
.plt	Plot file
.op2	Output2 file
.xdb	XDB file

NX Nastran забезпечує пряму підтримку інтерфейсів для інших продуктів, таких як NX Nastran Access та I-DEAS Master FEM. Для MSC.Patran підтримка

здійснюється через модуль DBC, який створює «графіку» бази даних. Графічну базу даних, створену модулем DBC, також може читати NX Nastran Access бібліотека об'єктних процедур. Цю бібліотеку об’єктів можна пов’язати зі створеною користувачем програмою, яка витягує дані з бази даних. Це непрямий спосіб для сторонніх постачальників отримати модель NX Nastran та інформацію про результати. Зв’язок з іншими програмними пакетами, такими як I-DEAS Master FEM, здійснюється через NX Модуль Nastran OUTPUT2, який створює читабельний файл FORTRAN, який конвертується цими зовнішніми програмами до власних форматів даних.

1.4 Ansys Mechanical та Finite Element Analysis (FEA)

Ansys Mechanical – це один з найкращих у своєму класі розв’язувач скінченними елементами та можливостями для покращення моделювання [6].

Ansys Mechanical дозволяє розв’язувати складні інженерні проблеми при створенні конструкцій і приймати кращі та швидші проєктні рішення. За допомогою інструментарію аналізу скінченними елементами (FEA), доступними у наборі, можна налаштовувати та автоматизувати розв’язання задач будівельної механіки та параметризувати їх для аналізу кількох сценаріїв проєктування. Ansys Mechanical – це динамічний інструмент, який має повний набір інструментів аналізу.

Ansys Mechanical створює інтегровану платформу, яка використовує скінченно-елементний метод (FEA) для структурного аналізу. Mechanical – це динамічне середовище, яке має повний набір інструментів аналізу, від підготовки геометрії для аналізу до підключення додаткової фізики для ще більшої точності. Інтуїтивно зрозумілий і настроюваний інтерфейс користувача дозволяє інженерам усіх рівнів швидко та з упевненістю отримувати відповіді.

Ansys Workbench забезпечує надійне підключення до комерційних інструментів САПР.

Динамічний аналіз дозволяє використовувати розширені параметри розв’язувача для різноманітних матеріалів і функцій:

– лінійна динаміка;
– нелінійності;
– термічний аналіз;
– матеріали;
– композити;
– гідродинаміка;
– взаємодія рідина-структура;
– налаштування та створення сценаріїв;
– управління рішеннями;
– високопродуктивні обчислення.

1.5 Rocky DEM та структурний аналіз

Процедура проєктування являє собою складний процес, і виникає питання з чого починати й у якому напрямку рухатися. Нехай потрібно зпроєктувати ковшовий конвеєр [9]. Як підійти до такого проєкту?

Історично склалося так, що інженери починали з відомого проєкту, проводили ручні розрахунки, робили припущення та проводили польові випробування. Ймовірно, ці конструкції провалили своє перше випробування. Отже, інженери повторили і спробували знову.

Створення фізичного прототипу потребує багато часу, коштів і зусиль. Як наслідок, це не сприяє циклам запуску конкурентоспроможних продуктів. Отже, інженери використовують високоточні інструменти моделювання (наприклад, аналіз методом скінченних елементів, методи обчислювальної гідродинаміки, метод дискретних елементів (DEM)) для проєктування продуктів.

Щоб розробити цей ковшовий конвеєр, інженери могли б об’єднати Ansys Mechanical і Rocky DEM для віртуального моделювання та оптимізації.

Інженери використовують програмне забезпечення FEA, наприклад Mechanical, для виконання структурного моделювання для цивільного, автомобільного, авіаційного та інших секторів.

Статичне моделювання визначає умови рівноваги та деформації конструкції під заданими навантаженнями. Для моделювання перехідних процесів умови рівноваги враховують як деформацію, так і кінематичну енергію. DEM є невід'ємним інструментом для вивчення динаміки часток. Він обробляє сипучі матеріали, такі як каміння, ґрунт, порошкоподібні хімічні речовини, харчові чіпси та фармацевтичні таблетки.

DEM враховує всі сили, що діють на кожну частинку всередині об'ємної системи. Потім він дає уявлення про те, як ці матеріали працюватимуть у певному компоненті в різних умовах процесу.

Rocky DEM може моделювати системи з багатьма частинками складної форми та точних розмірів. Інструмент використовується в багатьох галузях промисловості, зокрема:

– майнінг;
– важка техніка;
– сільське господарство;
– хімічний;
– фармацевтичний.

Під час моделювання Rocky DEM відстежує навантаження на кожен вузол геометричної сітки. Потім ці навантаження експортуються як поле тиску для подальшого аналізу за допомогою Mechanical. Потім програмне забезпечення FEA дискретизує геометрію та розв'язує статичні задачі механіки.

Поєднуючи структурний аналіз із Rocky DEM, інженери можуть симулювати випадки перехідних процесів, враховуючи зміну геометрії та змінні в часі навантаження, що діють на різноманітні елементи об'єкту проєктування.

Крім того, Rocky DEM повністю інтегровано в Ansys Workbench, тому не потребує додаткового програмного забезпечення для поєднання його з Mechanical. Це також дозволяє інженерам легко застосовувати інструменти

дослідження дизайну для віртуальних параметричних досліджень, оптимізації, аналізу надійності та створення поверхні відгуку.

Rocky DEM може відтворювати складні рухи в своєму інтерфейсі користувача, включаючи комбінований рух і вільний рух тіла, спричинений частинками, із 6 ступенями свободи.

Багато клієнтів поєднали програмне забезпечення Ansys із Rocky DEM, щоб покращити своє обладнання та процеси.

Наприклад, один із найбільших виробників залізної руди стикався з низькою ефективністю виробництва щоразу, коли подрібнена руда забивала рухомі сита біля основи бункерів. Це збільшило час обслуговування та простою для очищення обладнання.

Використовуючи Mechanical і Rocky DEM, компанія точно відобразила навантаження після звичайних процесів. Rocky DEM зафіксував широкий розподіл розміру та форми вхідної руди.

Це дозволило компанії впровадити ефективні зміни конструкції обладнання, такі як оптимізація кута нахилу, швидкості обертання, відстані та профілю роликових дисків. Після цих змін виробництво зросло на 11,4 %, що заощадило компанії 100 мільйонів доларів лише за 3 місяці.

Отже вибір правильних програмних інструментів дослідження виробничих процесів з метою вдосконалення існуючого обладнання або створення принципово нового має велике значення, й потребує створення нових програмних продуктів із новими можливостями.

2 МАТРИЦЯ ЖОРСТКОСТІ ПЛОСКОГО ТРИКУТНОГО СКІНЧЕННОГО ЕЛЕМЕНТУ

2.1 Постановка задачі

Розглянемо плоский трикутний скінченний елемент із координатами вузлів $i\left(x_1^{(i)}, x_2^{(i)}\right)$, j$(x_1^{(j)}, x_2^{(j)})$ і k$(x_1^{(k)}, x_2^{(k)})$. Кожен вузол має дві ступені свободи – можливість переміщень за двома напрямками x_1 і x_2 (рис. 2.1).

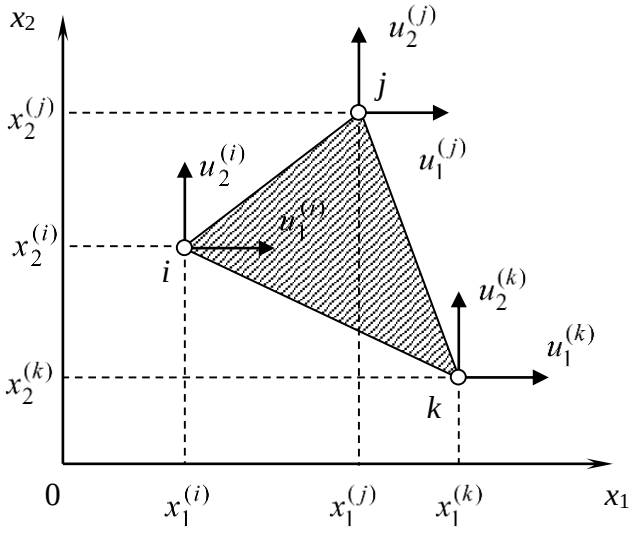

Рисунок 2.1 – Плоский трикутний скінченний елемент

Вектор переміщень вузлів скінченного елемента представимо у вигляді:

$$\{u\} = \begin{Bmatrix} u_1^{(i)} \\ u_2^{(i)} \\ u_1^{(j)} \\ u_2^{(j)} \\ u_1^{(k)} \\ u_2^{(k)} \end{Bmatrix}.$$

2.2 Розв’язання задачі

Припустимо, що функція переміщень $u_1(x_1, x_2)$ і $u_2(x_1, x_2)$ у межах скінченного елемента підпорядковується лінійному закону:

$$u_1(x_1, x_2) = \alpha_1 + \alpha_2 x_1 + \alpha_3 x_2, \quad (2.1)$$

$$u_2(x_1, x_2) = \alpha_4 + \alpha_5 x_1 + \alpha_6 x_2. \quad (2.2)$$

Шість сталих $\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5, \alpha_6$ можуть бути знайдені через вузлові переміщення та координати вузлів:

$$\alpha_1 = \frac{\Delta\alpha_1}{\Delta} = \frac{u_1^{(i)} a_1 + u_1^{(j)} a_2 + u_1^{(k)} a_3}{a_1 + a_2 + a_3},$$

$$\alpha_2 = \frac{\Delta\alpha_2}{\Delta} = \frac{u_1^{(i)} b_1 + u_1^{(j)} b_2 + u_1^{(k)} b_3}{a_1 + a_2 + a_3},$$

$$\alpha_3 = \frac{\Delta\alpha_3}{\Delta} = \frac{u_1^{(i)} c_1 + u_1^{(j)} c_2 + u_1^{(k)} c_3}{a_1 + a_2 + a_3},$$

де $a_1 = x_1^{(j)} x_2^{(k)} - x_1^{(k)} x_2^{(j)}$, $a_2 = x_1^{(k)} x_2^{(i)} - x_1^{(i)} x_2^{(k)}$, $a_3 = x_1^{(i)} x_2^{(j)} - x_1^{(j)} x_2^{(i)}$, $b_1 = x_2^{(j)} - x_2^{(k)}$, $b_2 = x_2^{(k)} - x_2^{(i)}$, $b_1 = x_2^{(i)} - x_2^{(j)}$, $c_1 = x_1^{(k)} - x_1^{(j)}$, $c_2 = x_1^{(i)} - x_1^{(k)}$, $c_3 = x_1^{(j)} - x_1^{(i)}$.

Підставимо співвідношення для a_1, a_2, a_3 у вираз для переміщення (2.1) і (2.2), і перетворимо його до вигляду:

$$u_1(x_1, x_2) = \frac{u_1^{(i)} a_1 + u_1^{(j)} a_2 + u_1^{(k)} a_3}{a_1 + a_2 + a_3} +$$

$$+ \frac{u_1^{(i)} b_1 + u_1^{(j)} b_2 + u_1^{(k)} b_3}{a_1 + a_2 + a_3} x_1 + \frac{u_1^{(i)} c_1 + u_1^{(j)} c_2 + u_1^{(k)} c_3}{a_1 + a_2 + a_3} x_2 =$$

$$= u_1^{(i)} \frac{a_1 + b_1 x_1 + c_1 x_2}{a_1 + a_2 + a_3} + u_1^{(j)} \frac{a_2 + b_2 x_1 + c_2 x_2}{a_1 + a_2 + a_3} +$$

$$+u_1^{(k)}\frac{a_3+b_3x_1+c_3x_2}{a_1+a_2+a_3}=u_1^{(i)}N_1+u_1^{(j)}N_2+u_1^{(k)}N_3, \quad (2.3)$$

де

$$N_1=\frac{a_1+b_1x_1+c_1x_2}{a_1+a_2+a_3},$$

$$N_2=\frac{a_2+b_2x_1+c_2x_2}{a_1+a_2+a_3}, N_3=\frac{a_3+b_3x_1+c_3x_2}{a_1+a_2+a_3}. \quad (2.4)$$

Вираз для $u_1(x_1,x_2)$ представиться аналогічно (2.3) наступним чином:

$$u_1(x_1,x_2)=u_2^{(i)}N_1+u_2^{(j)}N_2+u_2^{(k)}N_3, \quad (2.5)$$

де N_1, N_2, N_3 – функції форми, що знаходяться за формулами (2.4).

Вираз (2.3), (2.5) в матричній формі запису можна представити як:

$$\{U\}=[N]\{u\}, \quad (2.6)$$

де $\{U\}=\begin{Bmatrix}u_1(x_1,x_2)\\u_2(x_1,x_2)\end{Bmatrix}$ – вектор переміщень довільної точки кінцевого елемента, $\{u\}$ – вектор вузлових переміщень вигляду

$$\{u\}=\begin{Bmatrix}u_1^{(i)}\\u_2^{(i)}\\u_1^{(j)}\\u_2^{(j)}\\u_1^{(k)}\\u_2^{(k)}\end{Bmatrix},$$

$$[N]=\begin{bmatrix}N_1 & 0 & N_2 & 0 & N_3 & 0\\0 & N_1 & 0 & N_2 & 0 & N_3\end{bmatrix}.$$

$[N]$ – матриця функцій форми, підібрана таким чином, щоб після її підстановки її вираз (2.6) отримати формули (2.3) і (2.5).

У плоскій задачі теорії пружності незалежними компонентами тензора напружень є нормальні напруження σ_{11}, σ_{22} і дотичне σ_{12}. Тензор деформацій також має три незалежних компоненти – лінійні деформації $\varepsilon_{11}, \varepsilon_{22}$ і кут зсуву γ_{12}. Запишемо перераховані компоненти в векторному вигляді:

$$\{\sigma\} = \begin{Bmatrix} \sigma_{11} \\ \sigma_{22} \\ \sigma_{12} \end{Bmatrix}, \{\varepsilon\} = \begin{Bmatrix} \varepsilon_{11} \\ \varepsilon_{22} \\ \gamma_{12} \end{Bmatrix}.$$

Рівняння рівноваги запишуться у вигляді:

$$\begin{cases} \dfrac{\partial \sigma_{11}}{\partial x_1} + \dfrac{\partial \sigma_{12}}{\partial x_2} + G_1 = 0, \\ \dfrac{\partial \sigma_{21}}{\partial x_1} + \dfrac{\partial \sigma_{22}}{\partial x_2} + G_2 = 0, \end{cases}$$

або в матричній формі запису $[\partial]\{\sigma\} + \{G\} = 0$, де

$$[\partial] = \begin{bmatrix} \dfrac{\partial}{\partial x_1} & 0 \\ 0 & \dfrac{\partial}{\partial x_2} \\ \dfrac{\partial}{\partial x_1} & \dfrac{\partial}{\partial x_1} \end{bmatrix}$$

– матриця диференціального виду.

Вектор зовнішнього навантаження:

$$\{G\} = \begin{Bmatrix} G_1 \\ G_2 \end{Bmatrix}.$$

Вектор переміщень представляється у вигляді:

$$\{u\} = \begin{Bmatrix} u_1 \\ u_2 \end{Bmatrix}.$$

Геометричні рівняння:

$$\varepsilon_{11} = \frac{\partial u_1}{\partial x_1}, \varepsilon_{22} = \frac{\partial u_2}{\partial x_2}, \gamma_{12} = \frac{\partial u_1}{\partial x_1} + \frac{\partial u_2}{\partial x_2},$$

або в матричному вигляді $\{\varepsilon\} = [\partial]\{u\}$.

Фізичні рівняння (закон Гука):

$$\sigma_{11} = (2\mu + \lambda)\varepsilon_{11} + \lambda\varepsilon_{22},$$
$$\sigma_{22} = \lambda\varepsilon_{11} + (2\mu + \lambda)\varepsilon_{22},$$
$$\sigma_{12} = \mu\gamma_{12}.$$

В матричній формі ці відношення запишуться у вигляді:

$$\{\sigma\} = [E]\{\varepsilon\},$$

де

$$[E] = \begin{bmatrix} 2\mu + \lambda & \lambda & 0 \\ \lambda & 2\mu + \lambda & 0 \\ 0 & 0 & \mu \end{bmatrix},$$

тут $[E]$ – матриця пружних постійних матеріала.

Коефіцієнти μ та λ виражаються через модуль пружності E і коефіцієнт Пуассона ν для плоскої деформації:

$$\mu = \frac{E}{2(1+v)}, \lambda = \frac{Ev}{(1+v)(1-2v)},$$

для плоского напруженого стану:

$$\mu = \frac{E}{2(1+v)}, \lambda = \frac{Ev}{(1-v^2)}.$$

Враховуючи наведені відношення матриця пружних постійних набуде вигляду:

– для плоскодеформованого стану:

$$[E] = \frac{Ev}{(1+v)(1-2v)} \begin{bmatrix} 1-v & v & 0 \\ v & 1-v & 0 \\ 0 & 0 & \frac{1-2v}{2} \end{bmatrix};$$

– для плосконапруженого стану:

$$[E] = \frac{E}{(1-v^2)} \begin{bmatrix} 1 & v & 0 \\ v & 1 & 0 \\ 0 & 0 & \frac{1-v}{2} \end{bmatrix}.$$

Враховуючи відношення Коші для трикутного скінченного елемента отримаємо:

$$\{\varepsilon\} = [\partial]\{u\} = [\partial][N]\{\delta\} = [B]\{\delta\},$$

де $[B] = [\partial][N]$ – матриця векторів деформацій від одиничних переміщень вузлів скінченного елемента.

Матриця жорсткості скінченного елемента дорівнюватиме:

$$[K]_{(e)} = t \iint\limits_{S} [B]_{(e)}^{T}[E][B]_{(e)} dS,$$

де t – товщина конструкції.

Глобальна матриця системи буде мати вигляд:

$$[\bar{K}] = \sum_{e=1}^{l} [i]_{(e)}^{T}[K]_{(e)}[i]_{(e)},$$

де l – загальна кількість скінченних елементів, $[i]_{(e)}$ – матриця інциденцій.

Компоненти напружено-деформованого стану досліджуваного об’єкту знаходяться із розв’язання системи лінійних алгебраїчних рівнянь такого виду:

$$[\bar{K}]\{u\} = \{P\}, \qquad (2.7)$$

тут $\{u\}$ – вектор вузлових переміщень конструкції, $\{P\}$ – вектор вузлових навантажень.

3 ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ТА РЕЗУЛЬТАТИ ДОСЛІДЖЕННЯ

3.1 Блок-схема та основний програмний код

Побудуємо загальну блок-схему, на основі якої буде створена САПР на основі методу скінченних елементів для розв’язання плоских задач теорії пружності:

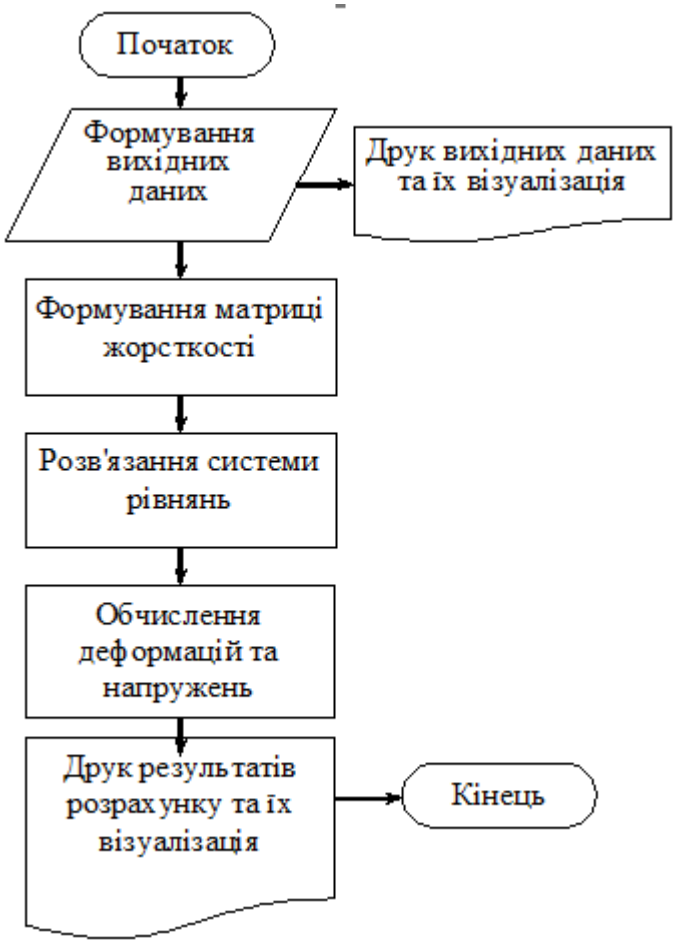

Рисунок 3.1 – Блок-схема алгоритму скінченно-елементного розв’язання плоских задач теорії пружності

Використовуємо бібліотеку numpy – це бібліотека Python, створена для обчислень та містить в собі багатовимірні масиви та матриці [10]. Також ця бібліотека містить в собі досить багато математичних та алгебраїчних функцій,

також функцій перетворень.

Numpy дуже тісно пов’язаний з pandas. Pandas є, свого роду, розширенням numpy [11].

Pandas – це пакет Python, який забезпечує швидкі, гнучкі та виразні структури даних, розроблені для того, щоб зробити роботу з «реляційними» або «міченими» даними одночасно легкою та інтуїтивно зрозумілою. Він має на меті стати основним будівельним блоком високого рівня для практичного аналізу реальних даних у Python. Крім того, він має ширшу мету – стати найпотужнішим і гнучким інструментом для аналізу/маніпулювання даними з відкритим кодом, доступним будь-якою мовою. Вона вже добре йде до цієї мети [12].

Фрагмент основного програмного коду для головної програми наведено в додатку А.

3.2 Результати дослідження

З використанням розробленої САПР проведемо розрахунок плоскої задачі механіки деформівного твердого тіла, а саме визначимо напружено-деформований стан консольної балки, защемленої з одного кінця і з силою прикладеною перпендикулярно до осі балки з іншого (див. рис. 3.2). Вважається, що балка знаходиться в умовах плоскої деформації.

Розміри консольної балки: довжина $b = 5$ см, ширина см, висота, $t = 0{,}1$ см, сила, прикладена на кінці балки, дорівнює 0,0008 МПа, модуль пружності матеріалу балки МПа коефіцієнт Пуассона $\upsilon = 0{,}35$.

За допомогою створеної САПР задаємо геометрію конструкції, крайові умови – защемлення на одному кінці балки та силу у напрямку осі *x* на іншому, а також пружні константи, що характеризують матеріал балки – модуль пружності та коефіцієнт Пуассона. Після цього постановка задачі для розрахунку є виконаною.

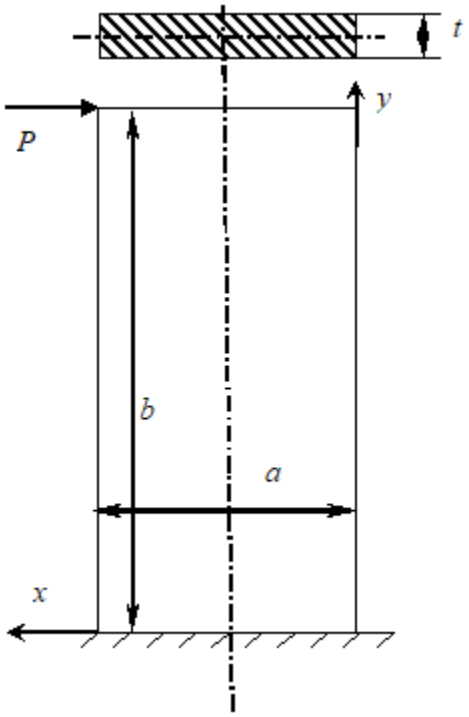

Рисунок 3.2 – Розрахункова схема вигину консольної балки силою, прикладеною на кінці

Для розв’язання методом скінченних елементів потрібно охарактеризувати тип скінченного елементу та сітку дискретизації на скінченні елементи. На даному етапі у САПР реалізовано один тип скінченного елементу – трикутний елемент із лінійною апроксимацією шуканої функції. Сітка дискретизації показує кількість скінченних елементів, на які розбивається конструкція, це робиться САПР автоматично згідно реалізованого закону дискретизації. Розрахунки проводились при різних сітках дискретизації до отримання певної збіжності результатів, тобто коли збільшення кількості скінченних елементів практично не впливає на результат розрахунку.

Результатом розв’язання системи (2.7) є значний масив числових даних – дискретних значень шуканої величини, у даному випадку – значень вектору вузлових переміщень $\{u\}$. Використовуючи формули наведені у попередньому розділі за цим вектором вузлових переміщень $\{u\}$ обчислюються компоненти тензора деформацій та компоненти тензора напружень.

Тобто в результаті проведеного розв’язання будемо мати великі масиви вузлових значень переміщень, деформацій та напружень. Для того щоб

використати отримані результати для подальшого аналізу потрібно вирішити основні дві проблеми:

– перевірка адекватності та точності отриманих результатів, відповідність сенсу поставленої задачі;

– можливість отримання додаткової інформації на основі результатів розв’язків.

Як правило цей останній етап за трудомісткістю та витраченим часом значно перевищує два попередні – виконання постановки задачі (хоча в залежності від специфіки задачі цей процес теж може бути не простий) і розв’язання задачі (цей процес, як правило повністю покладено на роботу ЕОМ). Тому для сучасних САПР є дуже актуальною проблема побудови третьої складової системи проєктування – постпроцесора – що дозволить вивішити вище оговорені задачі. Проблема аналізу великих масивів числових даних полягає у неможливості охопити загальну картину розв’язання задачі з усіма наслідками, що з цього випливають. Одним із найпоширеніших способів подолання цієї проблеми є візуальне представлення отриманих числових масивів, тобто за допомогою певних графічних об’єктів, це дозволяє відразу оцінити адекватність отриманих результатів й провести у подальшому їх детальний аналіз.

Візуальне представлення результатів чисельних розрахунків можливе за допомогою різноманітних графічних об’єктів, таких як графіки, двовимірні та тривимірні зображення, діаграми, лінії рівня тощо. У нашому випадку більш доцільним є прив’язка шуканої функції до геометричної моделі об’єкта. Одним із таких способів графічного представлення числових даних є напівтонове або кольорове зображення розподілу, де певний відтінок або колір прив’язаний до деякого діапазону числових значень.

Як правило, досліджувані об’єкти представляють собою тіла складної геометричної форми, але в результаті застосування методу скінченних елементів уся видима поверхня об’єкта дискретизується на елементи трикутної або чотирикутної форми, у характерних точках яких нам відомі значення шуканої функції. Тому для нас стоїть завдання певним чином записати закон розподілу

числової величини по трикутному (у нашому випадку) або чотирикутному елементу за значеннями цієї величини у кутових точках.

Існує певна кількість математичних моделей графічного представлення неперервної величини за її дискретними значеннями у вигляді певних кольорів та відтінків.

Одним із таких способів є побудова напівтонового зображення розподілу деякої величини функції u по трикутній області. З діапазону дискретних числових значень функції u знаходяться мінімальне $u_{\min}$ та максимальне $u_{\max}$ значення. Після чого обирається кількість градацій напівтону або кольору n для трикутної області. Діапазон значень u, який буде відповідати одному напівтону або кольору, матиме такий вид $[u_i, u_{i+1}]$, де $u_i = u_{\min} + ih$, $h = \dfrac{u_{\max} - u_{\min}}{n-1}$.

А номер напівтону або кольору для певного значення u^* можна знайти із залежності:

$$j = \frac{u^* - u_{\min}}{h}. \qquad (3.1)$$

Відобразимо напівтонове зображення для трикутника $P_1P_2P_3$ з дискретними значеннями шуканої функції – u_{P_1}, u_{P_2}, u_{P_3}. При чому точки пронумеруємо так, щоб u_{P_1} було максимальним дискретним значенням, а u_{P_2} – мінімальним. Тоді m_{12} – кількість напівтонових градацій на відрізку P_1P_2 – буде максимальною, й буде виконуватися рівність $m_{12} = m_{13} + m_{32}$:

$$m_{12} = \operatorname{int}\left(\frac{u_{P_1} - u_{\min}}{h}\right) - \operatorname{int}\left(\frac{u_{P_2} - u_{\min}}{h}\right);$$

$$m_{13} = \operatorname{int}\left(\frac{u_{P_1} - u_{\min}}{h}\right) - \operatorname{int}\left(\frac{u_{P_3} - u_{\min}}{h}\right);$$

$$m_{32} = \mathrm{int}\left(\frac{u_{P_3} - u_{\min}}{h}\right) - \mathrm{int}\left(\frac{u_{P_2} - u_{\min}}{h}\right), \qquad (3.2)$$

тут $\mathrm{int}(u)$ – ціла частина числа u.

Згідно цього способу графічне представлення деякої функції на трикутнику буде складатися із зображень деякої сукупності з m_{12} чотирикутників та трикутників, напівтон яких буде відповідати певному діапазону числових значень.

Такий підхід, реалізований у розробленій САПР для візуального представлення результатів чисельного розрахунку. На наведених нижче рисунках 3.3–3.7 представлено розподіл компонентів напружено-деформованого стану консольної балки, защемленої з одного кінця і з зосередженою силою, прикладеною до іншого кінця.

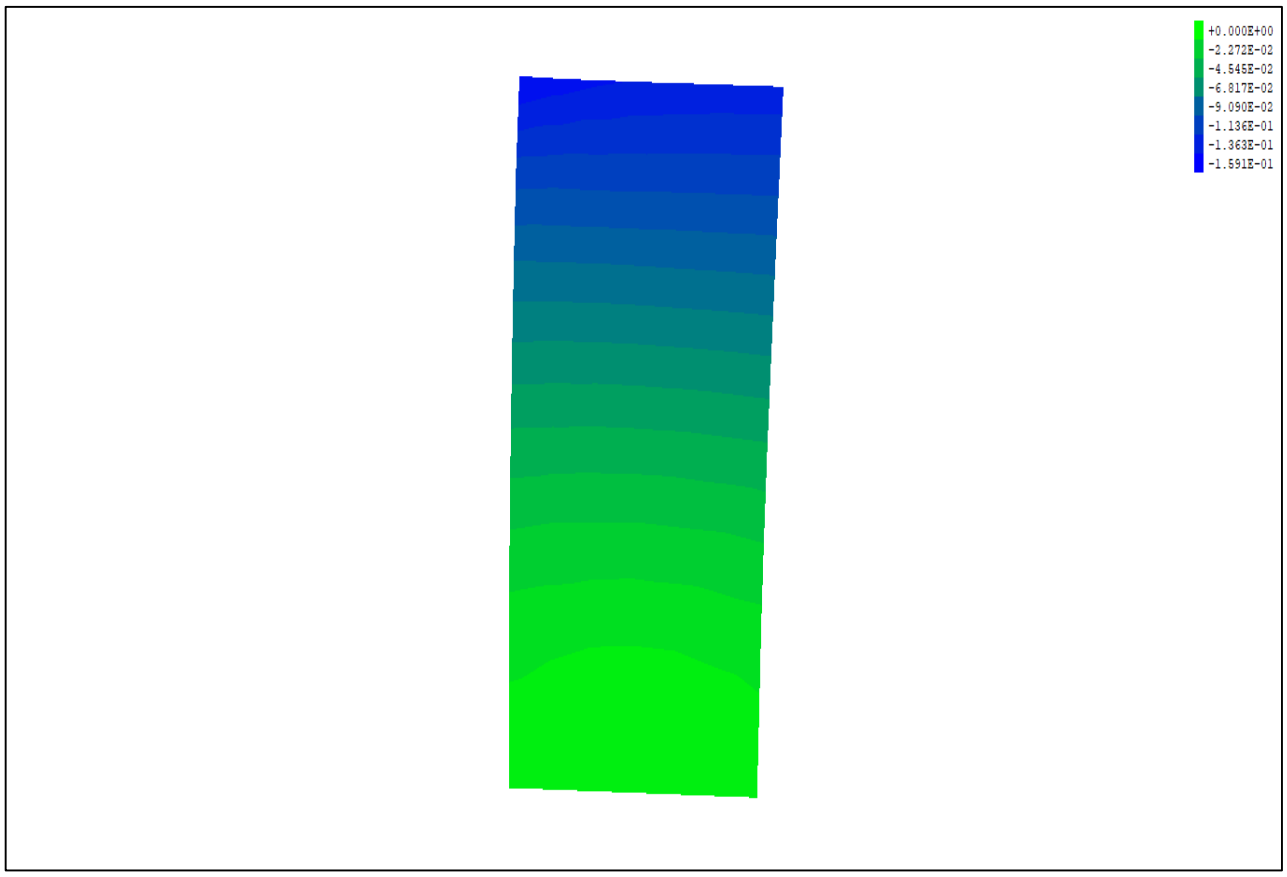

Рисунок 3.3 – Переміщення u_x

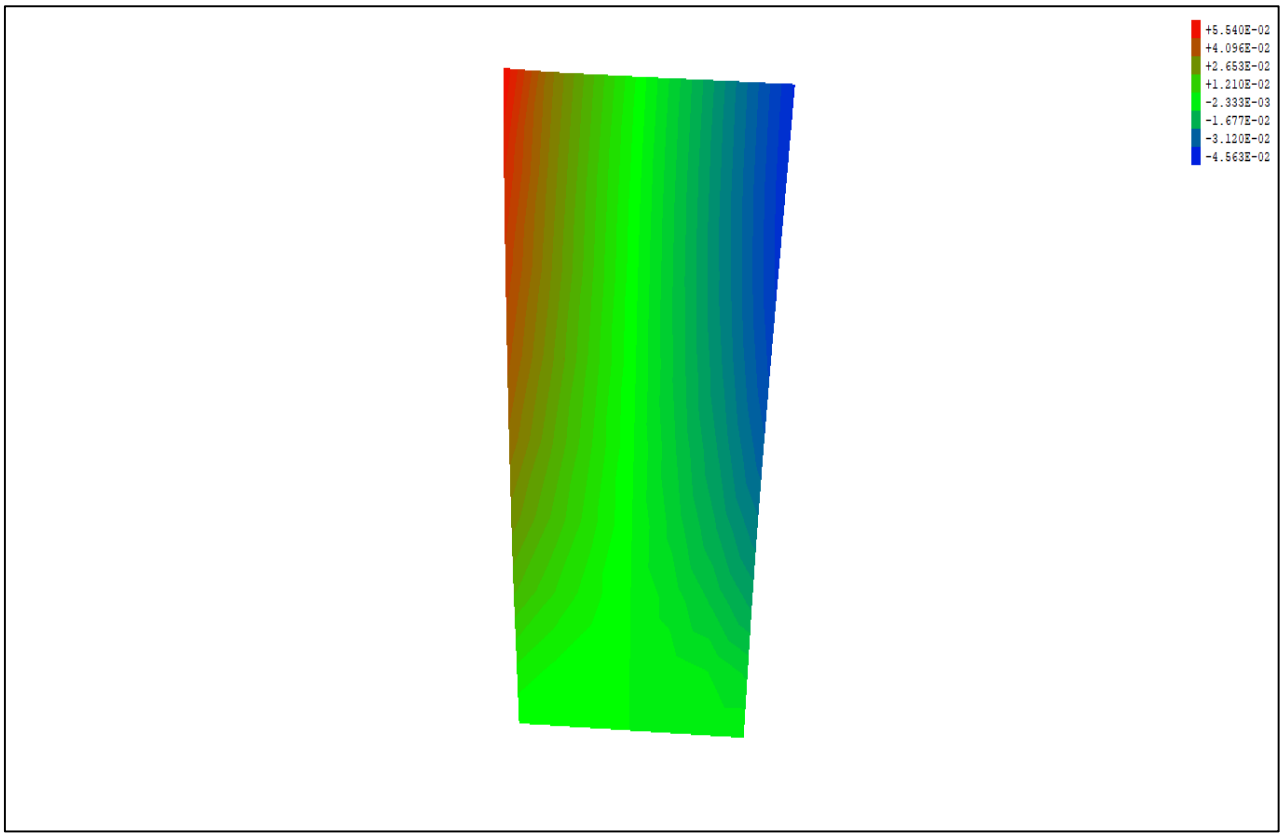

Рисунок 3.4 – Переміщення u_y

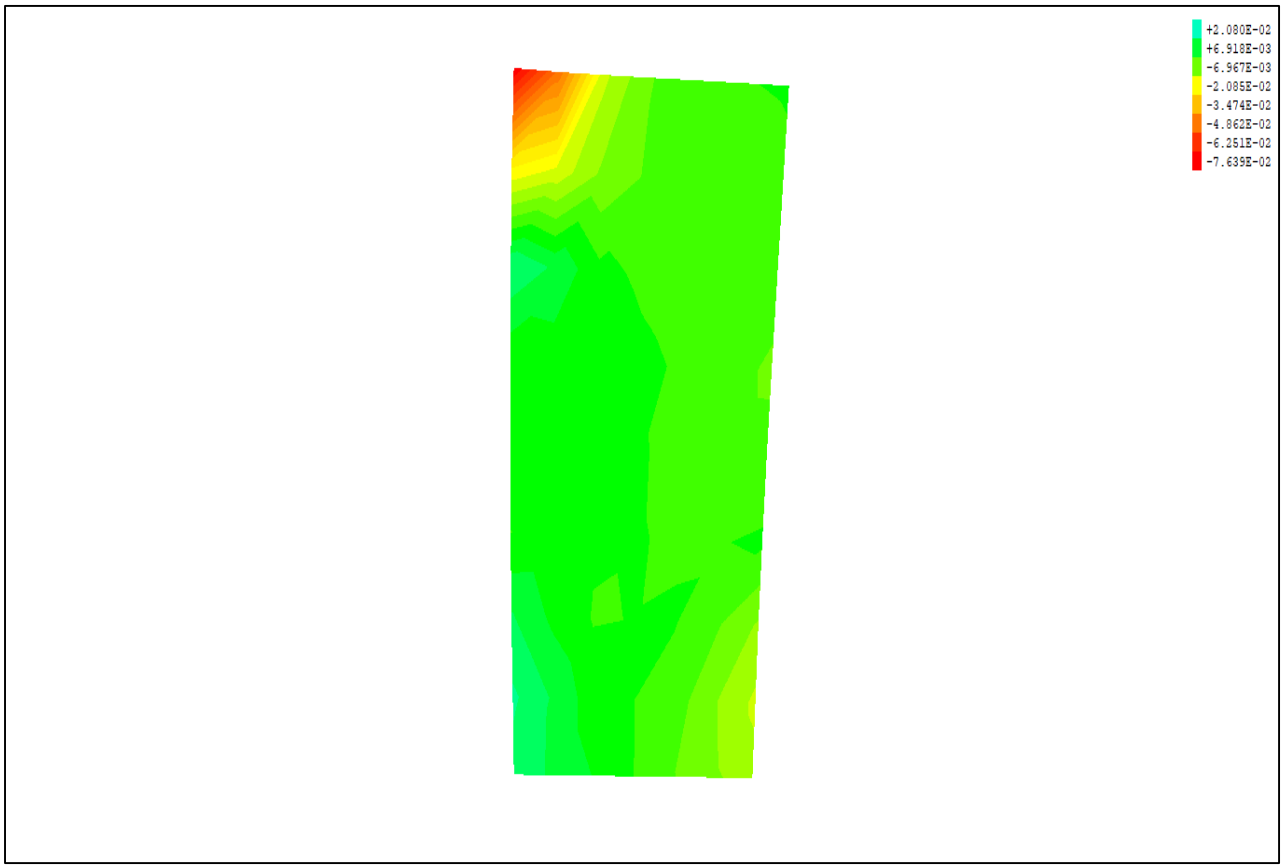

Рисунок 3.5 – Напруження σ_{xx}

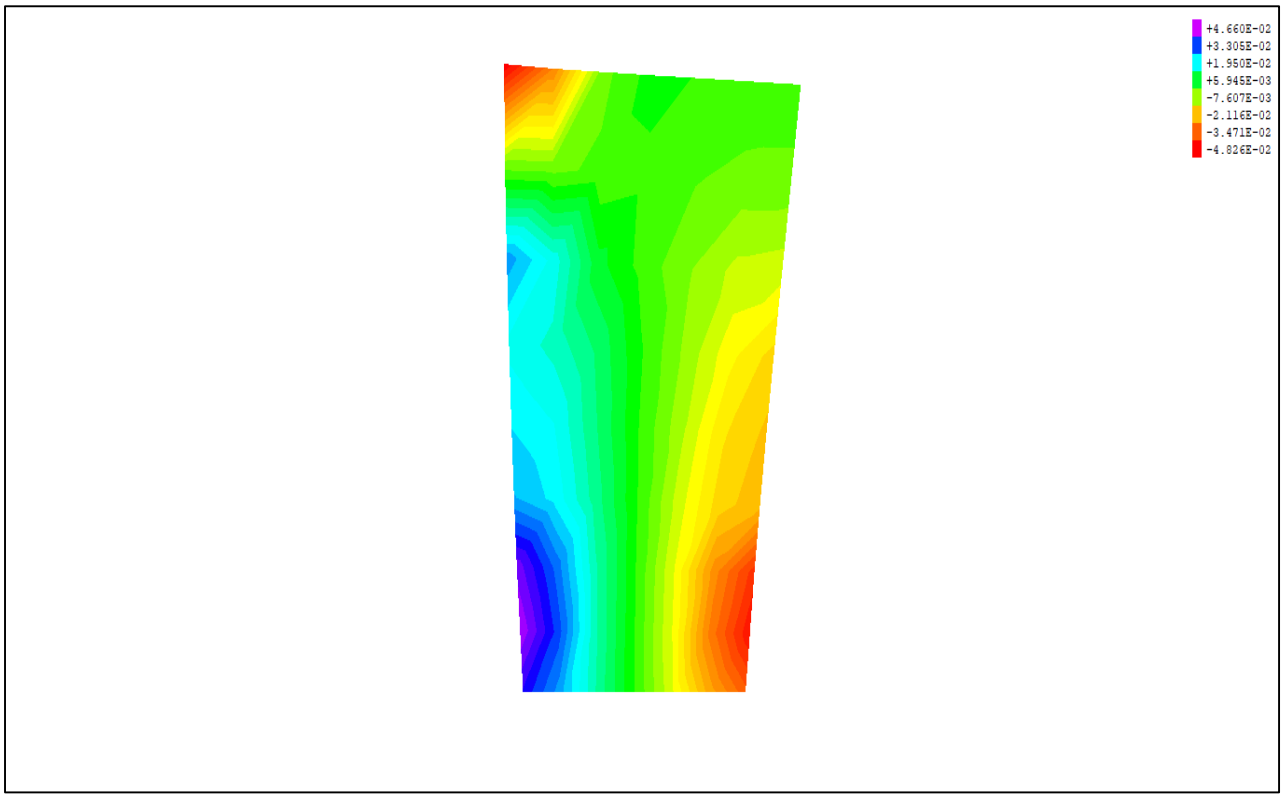

Рисунок 3.6 – Напруження σ_{yy}

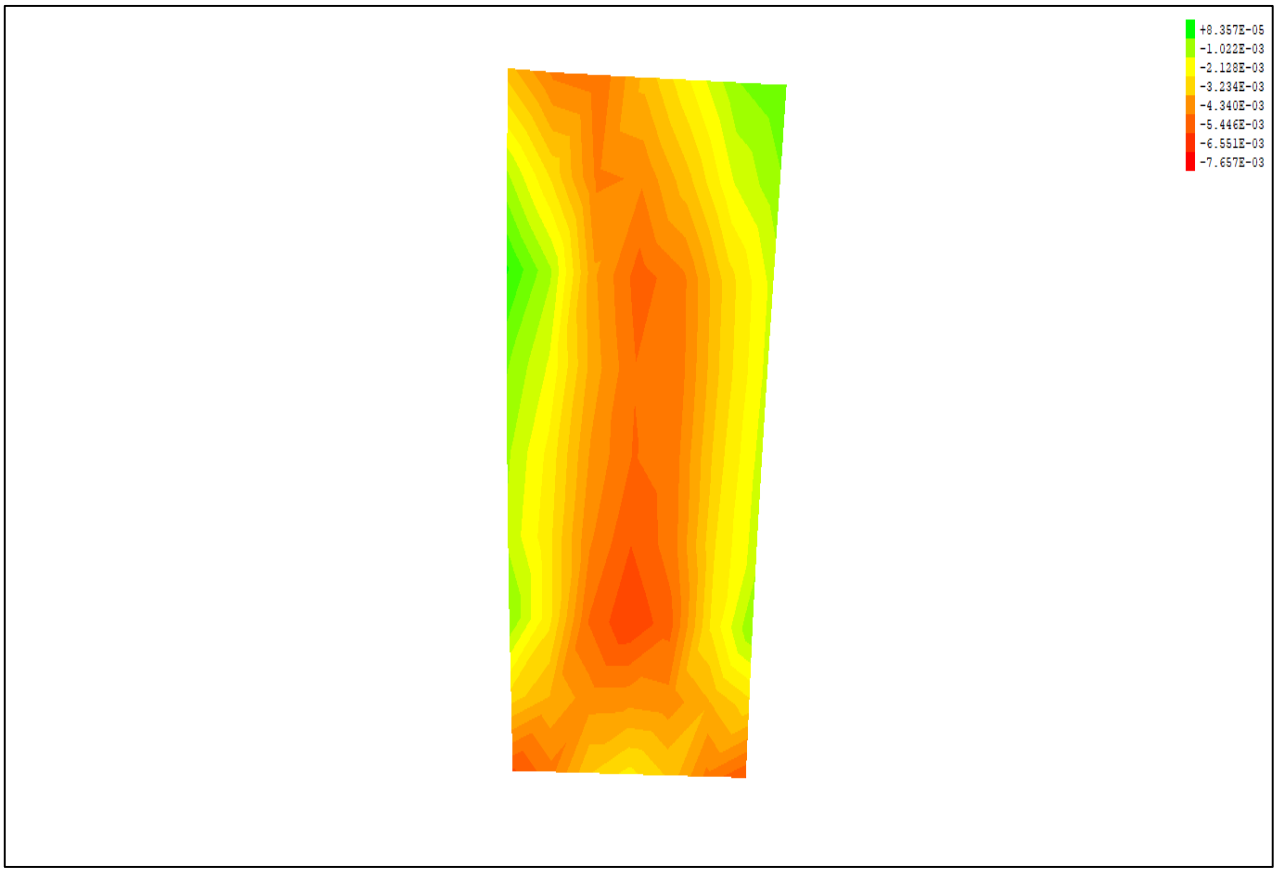

Рисунок 3.7 – Напруження σ_{xy}

Таким чином, продемонстровано можливості створеної САПР на прикладі задання вихідних даних, розрахунку та представлення результатів у задачі визначення напружено-деформованого стану консольної балки, защемленої з одного кінця і з силою прикладеною перпендикулярно до осі балки з іншого.

ВИСНОВКИ

Метод скінченних елементів широко використовується в різних сферах діяльності. У роботі проведено огляд сучасних САПР, що базуються на методі скінченних елементів, оцінено їх недоліки та переваги.

Існує ряд САПР які побудовані саме на розрахунках методів скінченних елементів, наприклад NASTRAN (NASA Structrual Analysis), Ansys Mechanical.

Розроблено структуру САПР для визначення напружено-деформованого стану двовимірних конструкцій. Наведено основні співвідношення для матриці жорсткості довільного трикутного скінченного елементу.

На базі мови програмування Python реалізована розроблена САПР.

Python являється високорівневою мовою програмування, що широко використовується в інтернет-додатках, розробці програмного забезпечення, науці даних та машинному навчанні. Має вагомі переваги, такі як:

– базовий, лаконічний та зрозумілий синтаксис, що легкий в читанні;

– легке поєднання з іншими мовами програмування;

– обширну бібліотеку, завдяки якій можна вирішити майже будь-яку поставлену задачу;

– можливість переносити на різні операційні системи: Windows, MacOS, Linux і Unix.

Синтаксис Python є досить зрозумілим та простим у використанні, при цьому, завдяки цій мові програмування дійсно можна вирішити багато задач.

Існують бібліотеки за допомогою яких можна розв’язувати та вирішувати математичні та інженерні задачі такі як NumPy, pandas. Саме ці бібліотеки були використані у роботі для матриці жорсткості плоского трикутного скінченного елементу. Також Python можна поєднувати з такими мовами програмування як Fortrun, C/C++.

За допомогою САПР визначено напруження та деформації консольної балки під дією сили, прикладеної на кінці.

ПЕРЕЛІК ПОСИЛАНЬ

1. Овчаренко В., Подлєсний С., Зінченко С. Основи методу кінцевих елементів і його застосування в інженерних розрахунках. Краматорськ : Донбас. держ. машинобуд. акад., 2008. 300 с.
2. Варак П. М., Бузун И. М., Городецкий А. С. Метод конечных элементов. Киев : Вища шк., 1981. 176 с.
3. Івченко І. Ю. Математичне програмування : навч. посіб. для студентів вищ. навч. закл. Київ : Центр учб. літ., 2007. 232 с.
4. Hernandez Ramos J. A. H. R., Tamarit M. A. R. Advanced Programming for Numerical Calculations : Climbing Python & Fortran. Madrid : UPM, 2022. 248 p.
5. MSC/NASTRAN for Windows user's guide. Los Angeles : MSC, 1999.
6. Ansys Mechanical. Structural FEA Analysis Software. Ansys. Engineering Simulation Software.URL: https://www.ansys.com/products/structures/ansys-mechanical#tab1-3 (дата звернення: 27.09.2022).
7. Ansys HFSS. 3D High Frequency Simulation Software. Ansys. Engineering Simulation Software. URL: https://www.ansys.com/products/electronics/ansys-hfss (дата звернення: 27.09.2022).
8. Ansys SIwave. Signal Integrity Analysis for PCB Design. Ansys. Engineering Simulation Software. URL: https://www.ansys.com/products/electronics/ansys-siwave (дата звернення: 05.10.2022).
9. Saurabh Sarkar. How to Apply Rocky DEM to Generate More Accurate Structural Analysis. Ansys. Ansys. Engineering Simulation Software. URL: https://www.ansys.com/blog/rocky-dem-structural-analysis (дата звернення: 10.10.2022).
10. Think Python: How to Think Like a Computer Scientist. Sebastopol. USA : O’Reilly Media, 2012. 300 p.
11. Scopatz A., Huff K. D. Effective Computation in Physics : Field Guide to Research with Python. O'Reilly Media, 2015. 552 p.

12. Мова програмування Python для інженерів і науковців : навчальний посібник. Івано-Франківськ : ІФНТУНГ, 2019. 275 с.

ДОДАТОК А

Фрагмент основного програмного коду для головної програми

```
import numpy as np
from numpy import *

class DIM:

    def __init__(self):
        self.M2 = M1()
        self.M2 = M1()

class RAZ:

    def __init__(self):
        self.BB = AA()
        self.BB = AA()
        self.CC = AA()
        self.DD = AA()

class Globals:
    dim = DIM()
    raz = RAZ()


    UPR = []
    TT = []
    AK = []
    AG = []
```

```
    Q = []
    AGP = []
    QP = []
    UD = []
    XX = []
    raz.AA = []
    raz.BB = []
    raz.CC = []
    DD = []

    fopen(4, FILE = "DATA.TXT", STATUS = "NEW")
    fopen(8, STATUS = "UNKNOWN", FORM = "BINARY", ACCESS =
"DIRECT", RECL = 7288)
    NUX = 100
    DAN(UPR, TT, Q, XX, NP, NUX)

    I = 1
    J = 1
    for I in M1 - 1:
        for J  in M2 - 1:
            NG = I + M1*(J - 1)
            AA = XX(NG + 1, 1) - XX(NG, 1)
            BB = XX(NG + M1, 2) - XX(NG, 2)
            CC = XX(NG + 1, 1) + XX(NG, 1)
            DD = XX(NG + M1, 2) + XX(NG, 2)
            KOFM(TT, UPR, AK)
            KOFG(AK, NG, AG, NUX)
            for J in 2*M1*M2:
                    QP = Q(J)
```

```
      for I in 2*M1*M2:
              AGP(I, J) = AG(I, J)
              SLAY(NP, AGP, QP, UD, NUX)
              NAPR(UPR, UD, NUX)
```