ДВНЗ «Донецький національний технічний університет»
Факультет комп'ютерно-інформаційних технологій та автоматизації
(повне найменування інституту, назва факультету)
Кафедра прикладної математики та інформатики
(повна назва кафедри)

«До захисту допущено»
В.о. завідувача кафедри
Наталія МАСЛОВА
(підпис) (Ім'я та ПРІЗВИЩЕ)
"___"_____2023р.

Випускна кваліфікаційна робота

бакалавра
(освітній ступінь)
на тему «Моделювання роботи сенсору виявлення витоку водню»

зі спецчастиною: «Розробка програмних модулів для моделювання роботи сенсору та інтерфейсу дослідження»

Виконав : студент 4 курсу, групи КН-19
(шифр групи)

Спеціальності 122 Комп'ютерні науки
(код і назва спеціальності)
освітньої програми 122 Комп'ютерні науки

Кузьменко Микити Андрійовича
(прізвище та ініціали) (підпис)

Керівник доц. каф. ПМІ, к.ф.-м.н., доц. Олена ЛЮБИМЕНКО
(посада, науковий ступінь, вчене звання, Ім'я та ПРІЗВИЩЕ) (підпис)

Рецензент доц. каф. ЕТ, к.т.н., доц. Олександр ШТЕПА
(посада, науковий ступінь, вчене звання, Ім'я та ПРІЗВИЩЕ) (підпис)

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

Студент _____
(підпис)

Луцьк – 2023 р.

ДВНЗ «Донецький національний технічний університет»

Факультет комп'ютерно-інформаційних технологій та автоматизації
Кафедра прикладної математики та інформатики
Освітній ступінь бакалавр
Спеціальність 122 Комп'ютерні науки
(код і назва спеціальності)
Освітня програма 122 Комп'ютерні науки

ЗАТВЕРДЖУЮ:
В.о. завідувача кафедри
_____Наталія МАСЛОВА
"____" _____20___року

ЗАВДАННЯ

на випускну кваліфікаційну роботу студента

Кузьменка Микити Андрійовича
(прізвище, ім'я, по батькові)

1.Тема роботи: «Моделювання роботи сенсору виявлення витоку водню» зі спецчастиною: «Розробка програмних модулів для моделювання роботи сенсору та інтерфейсу дослідження»

керівник роботи Любименко О.М., к.ф.-м.н., доцент,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом від "05" травня 2023 року № 175
2.Строк подання студентом проекту (роботи) 11.06. 2023 року
3.Вихідні дані до роботи результати переддипломної практики
4.Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1.Огляд джерел з теорії модулювання сенсорів та інтерфейсів .
2. Проектування програмних модулів для моделювання роботи сенсору та інтерфейсу дослідження .
3. Опис моделі для моделювання роботи сенсору та інтерфейсу дослідження
4.Опис роботи розроблених програмних модулів для моделювання роботи сенсору та інтерфейсу дослідження. .
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
робота містить 9 рисунків, 15 слайдів презентації та інший графічний матеріал, у кількості, достатній для демонстрації сутності роботи

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-4	Доц. Любименко О.М.	05.05.2023	10.06.2023
Нормоконтролер	Назарова І.А., доцент каф. ПМІ		12.06.2023

7. Дата видачі завдання 5 травня 2023 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строквиконанняетапів проєкту (роботи)	Примітка
1	Огляд літератури з предметної області	25.04.2023 - 05.05.2023р	вик
2	Описання та створення математичної моделі	06.05.2023- 13.05.2023р	вик
3	Розробка модулів для симуляції роботи сенсора	14.05.2023 – 24.05.2023р	вик
4	Створення інтерфейсу для дослідження результатів	25.05.2023 – 28.05.2023р	вик
5	Тестування та аналіз результатів моделювання:	29.05.2023 – 05.06.2023р	вик
6	Оформлення пояснювальної записки	06.06.2023 – 11.06.2023р	вик
7	Нормоконтроль пояснювальної записки, її перевірка антиплагіатною системою	12.06.2023 – 25.06.2023р	вик
8	Захист роботи	26.06.2023р	

Студент
(підпис) Микита КУЗЬМЕНКО
(Ім'я та ПРІЗВИЩЕ)

Керівник проєкту (роботи)
(підпис) Олена ЛЮБИМЕНКО
(Ім'я та ПРІЗВИЩЕ)

АНОТАЦІЯ

Кузьменко Микита Андрійович. Моделювання роботи сенсору виявлення витоку водню. Спецчастина: Розробка програмних модулів для моделювання роботи сенсору та інтерфейсу дослідження / Випускна кваліфікаційна робота на здобуття освітнього ступеня «бакалавр» за спеціальністю 122 "Комп'ютерні науки". ДонНТУ, Луцьк, 2023.

Об'єктом дослідження є процес моделювання роботи сенсору контролю витоку водню в реальних умовах.

Предметом дослідження є математична модель для контролю витоку водню.

Мета роботи є розробка програмних модулів для моделювання роботи сенсору та створення інтерфейсу з виводом результатів дослідження.

Методи досліджень базуються на основних положеннях системного аналізу, аналітичних методів опрацювання інформації.

Наукова новизна полягає в створенні програмних модулів, які забезпечують стабільність обчислень, адаптацію до потреб користувача та супроводжуються відповідною документацією коду.

Практичне значення полягає в створенні графічного інтерфейсу користувача для візуалізації результатів моделювання та аналізу даних.

Ключові слова: моделювання роботи сенсору, Python, бібліотеки numpy, matplotlib, tkinter, графічний інтерфейс користувача, аналіз даних, візуалізація даних, документація коду.

ABSTRACT

Kuzmenko Mykyta Andriyovych. Modeling the operation of a hydrogen leak detection sensor. Special part: Development of software modules for sensor operation modeling and research interface / Final qualifying work for obtaining the educational degree "Bachelor" in specialty 122 "Computer Science". - DonNTU, Lutsk, 2023.

The object of the study is the process of modeling the operation of the hydrogen leak control sensor in real conditions.

The subject of the study is a mathematical model for hydrogen leakage control.

The purpose of the work is the development of software modules for simulating the operation of the sensor and creating an interface with the output of research results.

Research methods are based on the main principles of system analysis, analytical methods of information processing.

Scientific novelty consists in the creation of software modules that ensure the stability of calculations, adaptation to user needs and are accompanied by appropriate code documentation.

The practical value is to create a graphical user interface for visualization of simulation results and data analysis.

Keywords: sensor operation simulation, Python, numpy libraries, matplotlib, tkinter, graphical user interface, data analysis, data visualization, code documentation.

ЗМІСТ

Вступ 8
1 Аналіз сфери використання сенсорів водню 12
1.1 Огляд джерел з теорії модулювання сенсорів та інтерфейсів 12
1.2 Аналіз умов роботи сенсора по витоку водню та визначення системних вимог 14
1.3 Аналіз функціональності сенсору виявлення витоку водню 16
1.4 Аналіз існуючих програмних модулів 18
1.5 Дискусійні та невирішені аспекти теми 18
1.6 Висновки по розділу 1 20
2 Проектування програмних модулів для моделювання роботи сенсору та інтерфейсу дослідження 22
2.1 Опис алгоритму розв'язання системи рівнянь пружності для металів при контакті з воднем 22
2.2 Основні бібліотеки Python для реалізаціїї графічного інтерфейсу 24
2.3 Висновок по розділу 2 29
3 Опис моделі для моделювання роботи сенсору та інтерфейсу дослідження 31
3.1 Розроблена математична модель 31
3.2 Чисельні методи для математичного моделювання процесу контролю 35
3.3 Висновки по розділу 3 38
4 Опис роботи розроблених програмних модулів для моделювання роботи сенсору та інтерфейсу дослідження 40
4.1 Опис створеного інтерфейсу 40

4.2 Реалізація математичної моделі 42

4.3 Висновки по розділу 4 50

Висновки 52

Перелік використаних джерел 54

Додаток А Зауваження нормоконтролера 57

Додаток Б Програмний код 58

Додаток В Презентація 71

ВСТУП

Сучасний світ характеризується швидким розвитком технологій, які стають все більш взаємопов'язаними та інтегрованими в нашому повсякденному житті. Сенсори та інтерфейси відіграють важливу роль у цьому процесі, оскільки вони є основними компонентами для отримання та обробки інформації з різних джерел та систем. Від промислових застосувань до смартфонів та автомобільних систем, сенсори та інтерфейси допомагають нам збирати, аналізувати та використовувати дані для оптимізації рішень та поліпшення якості життя.

Однак, з розвитком новітніх технологій та все більших потреб на інтелектуальні системи, необхідно розглядати нові підходи до моделювання сенсорів та інтерфейсів. Спрямоване це дослідження на аналіз сучасних методів та технологій, які використовуються в цій галузі, а також на виявлення можливих напрямів для подальших досліджень та розвитку.

У рамках даної роботи розглядатимуться такі аспекти, як актуальність теми, мета та завдання дослідження, об'єкт та предмет дослідження, аналіз діяльності підприємства, системні вимоги, апаратне та програмне забезпечення, інформаційні потоки, функціональний опис вихідного об'єкту, аналіз аналогічних систем та програм, постановка задачі та вибір методів рішення. Крім того, буде проведено критичний огляд літературних джерел, що відноситься до теорії модулювання сенсорів, теорії інтерфейсів для дослідження сенсорів, а також дискусійні та невирішені аспекти теми. У висновках підсумуються основні результати дослідження та актуальні напрямки для подальшого розвитку сенсорів та інтерфейсів.

Результати дослідження будуть в нагоді розробникам, інженерам та науковцям для кращого розуміння сучасного стану моделювання сенсорів та інтерфейсів, а також сприяти відкриттю нових можливостей для інновацій та розвитку технологій. Сенсори та інтерфейси стають важливими в нашому

сучасному світі, тому поглиблене розуміння принципів їх роботи та розвитку може призвести до заснування радикальних та гнучких систем, які поліпшують наше життя.

Ця робота ставить акцент на різних аспектах моделювання сенсорів та інтерфейсів, але вона не має на меті бути вичерпною або остаточною. Однак, це має бути відправною точкою для подальших досліджень, розглядів та співпраці між різними учасниками галузі, з метою досягнення більшої прозорості та розуміння відповідних технологій.

З часом, інновації в сенсорах та інтерфейсах можуть вирішити проблеми енергоефективності, надійності та безпеки. Відкрити інші можливості у таких галузях як Інтернет речей, квантові технології, біосенсори та машинне навчання. Дослідження має важливе значення для науки та практики.

Подальше дослідження може охоплювати аналіз методів моделювання сенсорів та інтерфейсів, порівняння ефективності, розробку нових моделей та принципів роботи. Можуть бути розглянуті сенсори та інтерфейси в специфічних умовах або сценаріях, таких як космічні або глибоководні дослідження.

Створенню нових рішень сприятиме міждисциплінарна співпраця науковців різних галузей. Інвестиції в освіту та дослідження у галузі сенсорів та інтерфейсів є важливими для підтримання інновацій та сталого розвитку технологій у майбутньому.

Актуальність теми "Розробка програмних модулів для модулювання роботи сенсору та інтерфейсу" обумовлена багатьма факторами. Такими як прогресивний розвиток технологій, зростання обсягів даних, вимоги до ефективності та енергоефективності, кросплатформеність, безпека, надійність, потреба у фахівцях та підвищення конкурентоспроможності. Ці фактори вказують на необхідність проведення такого дослідження в умовах сьогодення.

Актуальність теми "Розробка програмних модулів для модулювання роботи сенсору та інтерфейсу" відповідає сучасним технологічним

тенденціям. Розробка нових модулів адаптує сенсори до різних платформ, сприяючи їх широкому застосуванню у різних галузях. Розгляд цієї теми розвиває наукову базу знань і сприяє підготовці фахівців, відповідаючи сучасним вимогам ринку праці.

Мета дослідження - розробка програмних модулів для моделювання роботи сенсору та інтерфейсу з наміром підвищення ефективності та адаптації до різних умов.

Об'єкт дослідження - сенсорні системи та процеси взаємодії через програмні інтерфейси.

Предмет дослідження - розробка програмних модулів для моделювання роботи сенсорів та інтерфейсів.

Завдання включають аналіз методів, вивчення потреб користувачів, розробку концепції, алгоритмів, програмних модулів, тестування, інтеграцію, документацію, аналіз ризиків та правових аспектів. Виконання цих завдань допоможе досягти мети дослідження та розробки ефективних програмних модулів, відповідаючи сучасним тенденціям та вимогам користувачів.

Завдання включають вивчення теоретичних основ, розробку алгоритмів, реалізацію програмних модулів, тестування, інтеграцію, документацію, аналіз ризиків, застосування у різних галузях, розвиток та модернізацію, а також розробку нових продуктів та стратегій маркетингу.

Виконання завдань дозволить розробити програмні модулі для моделювання роботи сенсорів, які відповідають високим стандартам та інтеграції з різними системами. Це відкриє можливості для різних галузей, вдосконалить сенсорні системи та приверне партнерів та інвесторів. Завершення дослідження допоможе забезпечити розвиток ефективних програмних модулів та покращення якості роботи сенсорних систем у різних сферах.

У першому розділі «аналіз проблеми» автор ретельно проаналізував оцінку продуктивності моделювання датчика. Це передбачає комплексне вивчення специфікацій об’єкта дослідження, поточних методів моделювання

та їхніх обмежень, а також усвідомлення основних проблем, які виникають під час розробки програмного забезпечення датчиків.

Автор зосереджується на створенні інтерфейсу у другому розділі. Для ефективного виконання цього завдання використовуються Python і пов’язані бібліотеки, такі як numpy, matplotlib і tkinter. Отриманий інтерфейс зручний і без збоїв.

Для розділу «розроблена модель та результати» автор розробив програмні модулі для моделювання роботи датчика з одночасним обміном результатами. Тут приділяється увага кожному аспекту розробки, включаючи документацію коду, обчислювальну стабільність і адаптивність системи до вимог користувача.

Для тих, хто займається розробкою програмного забезпечення та моделюванням реальних процесів, робота автора пропонує унікальні ідеї. Завдяки розробці програмних модулів та інтерфейсу для симуляції роботи датчиків запропоновані практичні рішення потенційних проблем. Усі результати та висновки, отримані в результаті цієї роботи, мають високу цінність.

1 АНАЛІЗ СФЕРИ ВИКОРИСТАННЯ СЕНСОРІВ ВОДНЮ

1.1 Огляд джерел з теорії модулювання сенсорів та інтерфейсів

У книзі [1] розглядаються основні поняття та застосування детекторів (сенсорів) газів для виявлення та ідентифікації різних газів, а також різні техніки моделювання сенсорів.

В монографії [2] розглядається принципи роботи електронних носіїв та їх застосування, обговорюючи методи моделювання сенсорів на основі хімічних реакцій і фізичних процесів, а також підходи до обробки даних із сенсорів.

В статті [3] автори аналізують різні методи моделювання сенсорів, зокрема нейронні мережі та статистичні методи, для вирішення конкретних завдань в області хімічного сенсингу.

Оглядовий матеріал [4] розглядає оптичні сенсори та їх використання для виявлення хімічних речовин та біомолекул, демонструючи різні методи моделювання оптичних сенсорів та їх застосування в різних сферах.

У огляді [5] автори зосереджуються на розробці та моделюванні мікроелектромеханічних систем (MEMS) для інерційних сенсорів, розглядаючи ключові аспекти розробки MEMS-сенсорів, включаючи матеріали, процеси виготовлення та моделювання динамічної поведінки сенсорів.

У статті [6] автори розглядають моделювання та симуляцію детекторів положення для вимірювання часу прольоту, аналізуючи методи моделювання сенсорів на основі їх електронних характеристик та фізичних властивостей.

Стаття [7] присвячена вивченню технологій зібрання енергії на основі п'єзоелектричних матеріалів, розглядаючи різні підходи до моделювання п'єзоелектричних сенсорів, їхніх матеріалів та механізмів роботи, а також потенційні застосування.

У огляді [8] автори зосереджуються на наноструктурованих матеріалах та пристроях для виявлення біомолекул, розглядаючи різні типи сенсорів, їхні матеріали, методи моделювання та можливі застосування.

У книзі [9] розглядаються принципи, проектування та застосування бездротових сенсорних мереж, зокрема різні інтерфейси для збору та обробки даних з сенсорів.

Книга [10] розглядає різні сенсорні технології та їх застосування в інтелектуальних транспортних системах, описуючи різні інтерфейси для комунікації між сенсорами та іншими компонентами системи.

У огляді [11] автори зосереджуються на гнучких електронних матеріалах та їх застосуванні у різних пристроях, включаючи сенсори та інтерфейси для їх дослідження.

У книзі [12] автори розглядають різні методи розробки інтелектуальних матеріалів та мікроелектромеханічних систем (MEMS), включаючи сенсори та інтерфейси для їх дослідження.

Книга [13] розглядає різні типи носимих та імплантабельних медичних пристроїв, включаючи сенсори та інтерфейси для їх дослідження та моніторингу.

В цілому, враховуючи аналіз різних джерел, можна сказати, що важливо розуміти специфіку роботи сенсорів, особливості їх застосування та інтерфейсів для дослідження та обробки даних. Різні сенсорні технології та матеріали мають свої переваги та недоліки, і вибір відповідного варіанту залежить від конкретних вимог та обмежень підприємства, а також від доступних ресурсів, технологій та часових рамок.

Вивчення передового досвіду та інноваційних рішень, що вже використовуються в даній сфері, може допомогти уникнути помилок, визначити найбільш ефективні технології та підходи для власної розробки, а також забезпечити успішне проєктування та впровадження ефективної інформаційної системи.

1.2 Аналіз умов роботи сенсора по витоку водню та визначення системних вимог

Сенсор водню - це пристрій, який використовується для виявлення та вимірювання концентрації водню в середовищі. Датчик по вимірюванню водню, являє собою тонка металева пластина, яка при взаємодії з воднем, насичується воднем, та в результаті чого в неї змінюються довжина та товщина, і вона замикає контакт, в результаті чого поступає сигнал про виток водню. Тому являє інтерес математично промоделювати процес зміни концентрації водню в результаті дифузії водню та зміни форми пластини. Ескіз такого сенсору наведена на рис. 1.1 [14].

Рисунок 1.1 – Сенсор по виявленню водню

Кінетику адсорбції та десорбції водню на тонких дротах паладію вивчали за допомогою термопружніх вимірювань та аналіз, оснований на фізиці процесу, підтверджує дисоціативну адсорбцію водню на паладії з подальшою молекулярною десорбцією [14]. Крім того цікаво, визначити напруження в матеріалі, концентрацію водню, величину зсуву пластини та величину її деформації при адсорбції воднем.

В роботі [14] було виміряно час відгуку та час відновлення датчика, а також досліджено та обговорено їх залежність від концентрації водню.

Завдяки своїй видатній селективності до водню паладієві мембрани привернули великий інтерес у дослідженнях і розробках.

Для аналізу розробки програмних модулів сенсора та інтерфейсу дослідження, слід враховувати [15]:

1. Галузь підприємства: визначення специфіки діяльності[16].
2. Системні вимоги: оцінка вимог до програмних модулів.
3. Апаратне та програмне забезпечення: розгляд сумісності з існуючими системами.
4. Інформаційні потоки: визначення методів обробки, передачі та зберігання даних.
5. Функціональний опис вихідного об'єкту: виявлення основних вимог до програмних модулів.
6. Аналіз аналогів: вивчення передового досвіду та інноваційних рішень.
7. Постановка задачі: формулювання завдань для розробки програмних модулів.
8. Обґрунтування вибраного напряму та методів рішення: врахування доступних ресурсів, технологій, часових рамок.

Детальний аналіз дасть можливість розробити ефективну інформаційну систему, що відповідатиме запитам підприємства та сучасним тенденціям розвитку технологій.

Визначаючи системні вимоги для програмних модулів моделювання роботи сенсорів та дослідницьких інтерфейсів, слід розглянути:

1. Швидкість: забезпечення швидкої роботи системи.
2. Точність: забезпечення правильності моделювання та аналізу даних.
3. Надійність: стала робота модулів без збоїв.
4. Безпека: захист даних від несанкціонованого доступу.
5. Сумісність: інтеграція з іншими системами та пристроями.
6. Масштабованість: перспектива розширення функціоналу.

7. Гнучкість: пристосування до різних сенсорів та вимог.
8. Інтеграція: об’єднання з існуючими системами.
9. Стандартизація: відповідність технічним стандартам.

З огляду на вимоги, можна створити ефективні програмні модулі та інтерфейси для сенсорів. Важливо аналізувати вимоги, та використовувати технічне завдання для детального опису вимог, архітектури та інших аспектів розробки. Процес розробки охоплює створення прототипів, тестування та збір зворотного зв'язку для забезпечення якості та відповідності програмних модулів.

Апаратне забезпечення [17]:

1. Мікроконтролери: основа сенсорних систем, опрацювання даних.
2. Сенсори: типи залежать від застосування, вимог до системи[18].
3. Інтерфейси зв'язку: передача даних між пристроями.

Програмне забезпечення:

1. ОС: різні операційні системи для програмних модулів та контролю.
2. Мови програмування: вибір залежить від задачі, сумісності, уподобань.
3. IDE: створення, тестування, налаштування коду.
4. Бібліотеки, фреймворки: спрощення розробки програмних модулів.
5. Симулятори, емулятори: імітація роботи системи.
6. Системи контролю версій: відслідковування змін у коді.
7. Інструменти автоматизації збірки, тестування: полегшення розробки.
8. Документація, стандарти: відповідність технічним вимогам.

1.3 Аналіз функціональності сенсору виявлення витоку водню

Універсальні програмні модулі для модулювання роботи сенсорів та інтерфейсу, зосереджені на полегшенні дослідження та аналізу даних від сенсорів по уловлюванню газів.

Головні аспекти функціональності містять в собі модулювання роботи сенсорів, обробку та аналіз даних, зручний інтерфейс дослідження,

адаптивність та масштабованість, інтеграцію з іншими системами, а також безпеку та надійність. Реалізація цих вимог дозволить створити дієвий інструмент для дослідження та аналізу роботи сенсорів, сприяючи розвитку технологій сенсорів у різних сферах науки та промисловості.

Аналіз інформаційних потоків дозволить отримати:

1. Обсяг інформаційних потоків: оцінювання кількості даних, що передаються та обробляються в системі [19].

2. Формат даних: виявлення структури та формату інформації для забезпечення сумісності між компонентами.

3. Швидкість передачі: оцінка пропускної здатності системи та вимог до швидкості передачі даних.

4. Надійність інформаційних потоків: забезпечення стабільності та відмово стійкості системи передачі даних.

5. Безпека інформаційних потоків: збереження даних від несанкціонованого доступу та зловживань.

6. Інтеграція інформаційних потоків: забезпечення можливості сумісності та взаємодії різних джерел інформації.

7. Оптимізація інформаційних потоків: використання методів стиснення даних, пріоритезації, адаптивного управління для покращення результативності системи.

8. Аналіз майбутніх потреб інформаційних потоків: врахування можливого розвитку системи для забезпечення стабільності та масштабованості.

Врахування вище наведених аспектів дозволить розробити радикальну та надійну систему моделювання роботи сенсорів та інтерфейсів дослідження, яка зможе відповідати сучасним вимогам та формуватися разом з прогресом технологій.

1.4 Аналіз існуючих програмних модулів

Розробка нових програмних модулів, які враховують можливі напрямки вдосконалення програмних модулів для модулювання роботи сенсорів та інтерфейсу їх дослідження базується на використанні програм, баз даних та систем [20] (LabVIEW, MATLAB, Simulink, OpenModelica, Python з бібліотеками SciPy, NumPy та pandas). Розглядаючи кожен з них можна виділити, що:

LabVIEW пропонує чималий набір інструментів для моделювання сенсорів та дослідження їх роботи. Та він може бути складним для освоєння новими користувачами.

MATLAB має потужні можливості для аналізу даних та моделювання сенсорів, але вимагає необхідних знань з програмування та математики.

Simulink надає графічний інтерфейс для моделювання динамічно.

OpenModelica дозволяє створювати складні моделі сенсорів та їх інтерфейсів, але можливі обмеження у графічному інтерфейсі та підтримці різних видів сенсорів.

Python з додатковими бібліотеками SciPy, NumPy та pandas може бути гнучким інструментом для створення програмних модулів, але вимагає від користувача знання цих бібліотек.

Отже, розробка нових програмних модулів, які враховують потреби дослідників, може сприяти покращенню процесу моделювання сенсорів та їх інтерфейсів дослідження. Це може включати розвиток інтуїтивних графічних інтерфейсів, підтримку різних типів сенсорів та забезпечення масштабованості та адаптивності для інших сценаріїв застосування.

1.5 Дискусійні та невирішені аспекти теми

При розгляді теми моделювання сенсорів та інтерфейсів для дослідження сенсорів, варто звернути увагу на деякі дискусійні та невирішені аспекти:

- Моделювання сенсорів у випадках змішаних або комплексних оточень. В реальних умовах сенсори нерідко вимірюють сигнали в складних та динамічних середовищах. Розробка точних та надійних моделей для таких ситуацій є викликом і вимагає додаткових досліджень.

- Мініатюризація сенсорів та інтерфейсів. Зменшення розміру сенсорів та інтерфейсних компонентів може призвести до зниження енергоспоживання та покращення мобільності системи. Однак мініатюризація може також спричинити зниження точності вимірювань та інших обмежень.

- Методи обробки даних з сенсорів. При використанні різних моделей сенсорів та інтерфейсів виникає потреба у розвитку нових методів обробки та аналізу даних, які дають можливість отримувати корисну інформацію з вимірювань.

- Безпека систем сенсорів. Це включає розробку безпечних та надійних протоколів комунікації, а також методів захисту від зовнішніх загроз, таких як хакерські атаки та впливи зовнішнього середовища.

- Енергоефективність сенсорів та інтерфейсів. У деяких застосуваннях, особливо в бездротових сенсорних мережах, енергоефективність є критичним фактором. Розробка енергоефективних моделей сенсорів та інтерфейсів вимагає додаткових досліджень та оптимізації конструкцій.

- Інтеграція різних типів сенсорів та інтерфейсів. В ряді застосувань може виникнути потреба в об’єднанні різних типів сенсорів та інтерфейсів для отримання більш повної інформації про об'єкт або середовище.

- Використання штучного інтелекту та машинного навчання для моделювання сенсорів може допомогти покращити точність та ефективність сенсорних систем.

- Впровадження Інтернету речей (IoT) та промислового Інтернету речей (IIoT) може сприяти більшій інтеграції та гнучкості систем.

- Використання квантових технологій у моделюванні сенсорів та інтерфейсів можуть допомогти створити нові, точніші та ефективніші системи.

- Робастність та стійкість сенсорів та інтерфейсів може покращити надійність та продовжити термін служби систем.

- Міждисциплінарність досліджень та розробок у сфері сенсорів та інтерфейсів. Через різноманітність застосувань сенсорів та інтерфейсів, це відкрає нові можливості та веде до інноваційних рішень у моделюванні сенсорів та інтерфейсів.

1.6 Висновки по розділу 1

У процесі розробки програмних модулів для модулювання роботи сенсорів та інтерфейсів дослідження, потрібно зважати на ряд апаратних та програмних факторів. Підвищення якості програмного забезпечення та забезпечення надійності сенсорної системи залежить від вибору апаратного та програмного забезпечення, яке відповідає потребам завдання, вимогам до системи та особистим інтересам розробника.

Розробка програмних модулів для модулювання роботи сенсору та інтерфейсу його дослідження є актуальною та перспективною темою для науково-дослідної роботи. Вона відкриває нові можливості для покращення ефективності роботи сенсорних систем, розширення застосування сенсорів у різних сферах науки та техніки, а також сприяє підготовці кваліфікованих фахівців та розвитку науково-технічного прогресу в цій області.

На основі проведеного аналізу, можна зробити наступні висновки:

1. Існують різні методи та підходи до моделювання сенсорів та інтерфейсів, однак деякі з них мають обмеження та вимагають подальших досліджень та опрацювання. Внаслідок проведеного критичного огляду літературних джерел, було виявлено, що не дивлячись на значний прогрес в галузі, є ряд дискусійних та відкритих аспектів, що відносяться до теорії модулювання сенсорів та інтерфейсів, таких як підвищення точності моделей, оптимізація алгоритмів, удосконалення інтерфейсів тощо.

2. Визначення системних вимог, апаратного та програмного забезпечення, аналіз інформаційних потоків та функціонального опису

вихідного об'єкту допомагають розкрити ключові аспекти проєктування інформаційної системи та обрати оптимальні методи та техніки розробки програмних модулів, сформувати чіткий план розробки програмних модулів для модулювання роботи сенсорів та інтерфейсу їх дослідження, зважаючи на специфіку підприємства та його запити.

З урахуванням зроблених висновків, подальші розділи роботи будуть направлені на розробку конкретних програмних модулів, які відповідають виявленим вимогам такі як точність моделювання, ефективність, надійність і стабільність, візуалізація результатів та специфіці дослідження, а також на практичне застосування розроблених програмних модулів для модулювання роботи сенсорів та інтерфейсу їх дослідження.

2 ПРОЕКТУВАННЯ ПРОГРАМНИХ МОДУЛІВ ДЛЯ МОДЕЛЮВАННЯ РОБОТИ СЕНСОРУ ТА ІНТЕРФЕЙСУ ДОСЛІДЖЕННЯ

2.1 Опис алгоритму розв'язання системи рівнянь пружності для металів при контакті з воднем

Для реалізації алгоритму розв'язання системи рівнянь водневої пружності використовували знайдені в літературі методи, а саме детально розглянули один з них описаний в роботі Сташука В.А. [21], який полягає в тому, що цей метод є ефективним чисельним підходом для аналізу властивостей пружних тіл та їх реакції на дію зовнішніх сил. Через послідовність виконання кроків алгоритму, врахування матеріальних характеристик тіла та використання критерію збіжності, метод Сташука дозволяє отримати точні та надійні результати.

Алгоритм програмної реалізації в роботі Сташука В.А. [21] передбачає задання початкових умов, визначення матеріальних властивостей тіла, обчислення матриць пружності, тензорів деформацій та напружень, а також перевірку збіжності та корекцію результатів. Застосування цього алгоритму відкриває можливості для різноманітних інженерних та наукових застосувань, пов'язаних з дослідженням пружних тіл та розробкою сенсорів та інтерфейсів для їх взаємодії.

При реалізації такого алгоритму треба:

1. Задати початкові константи, що стосуються водню, матеріалу та просторових та часових параметрів.
2. Перевірити стійкість методу: коефіцієнт дифузії, часовий крок і крок у просторі повинні відповідати умові $D * \Delta t / \Delta h ** 2 < 0.5$.
3. Ініціалізувати масиви для збереження концентрації водню, зміщень та напружень.
4. Встановлювати початкові умови для концентрації водню та зміщень.

5. Для кожного моменту часу та кожного вузла пластини проводити розрахунки нової концентрації водню та зміщення з урахуванням впливу попереднього вмісту водню на матеріал.

6. Оновлювати значення концентрації водню та зміщення на межах пластини.

7. Обчислювати деформації як похідна зміщень та напруження.

Також в роботі [22 - 23] описано як знати матриці пружності:

$$C11 = Y/(1-2v)\ C12 = Yv/(1-v)\ C44 = G,$$

де Y - модуль Юнга,

v - коефіцієнт Пуассона,

G - модуль зсуву.

Тензор деформацій:

$$eij = (ui,j + uj,i)/2$$

де ui,j - компоненти вектора зміщення в точці (i,j).

Тензор напружень [23]:

$$\sigma ij = Cijkl * ek,l,$$

де Cijkl - компоненти матриці пружності, ek,l - компоненти тензора деформацій.

Критерій збіжності:

$$|\sigma_new - \sigma_old|/|\sigma_new| < \varepsilon,$$

де σ_new - новий тензор напружень,

σ_old - попередній тензор напружень,

ε - задана точність.

Після вдалого виконання алгоритму [24] та отримання необхідної сумісності, результати можуть бути використані для оцінки властивостей пружних тіл [25], а також для аналізу їх реакції на дію зовнішніх сил [26]. Метод Стащука [21] є дійовим чисельним інструментом для вирішення таких задач, що забезпечує точні та надійні результати.

2.2 Основні бібліотеки Python для реалізаціїї графічного інтерфейсу

Після вдалого виконання алгоритму створюються графіки, що відображають зміну концентрації водню, зміщень, деформацій та напружень з часом і по товщині пластини.

Реалізується користувацький інтерфейс на основі Tkinter, який дозволяє вводити нові параметри та відображати результати у вигляді графіків.

Tkinter, PyQt, wxPython, Kivy та PyGTK - це різні бібліотеки Python, які використовуються для створення графічного інтерфейсу користувача (GUI). Кожна з них має свої переваги та недоліки, які варто враховувати при виборі бібліотеки для конкретного проекту.

Tkinter - це стандартна бібліотека Python для створення графічного інтерфейсу користувача (GUI). Вона є обгорткою над Tcl/Tk, популярною бібліотекою GUI, яка використовується в багатьох мовах програмування. Tkinter пропонує різноманітні віджети, такі як кнопки, меню, текстові поля, мітки та інше, для створення інтерактивних додатків з вікнами. Вона працює на багатьох платформах, включаючи Windows, Linux та MacOS.

Переваги Tkinter:

1. Стандартна бібліотека Python: Tkinter є стандартною бібліотекою Python, тому вам не потрібно встановлювати її окремо.

2. Простота використання: Tkinter досить легко вивчити та використовувати, особливо для початківців.

3. Функціональність: Tkinter має достатньо віджетів для створення простих GUI.

4. Підтримка крос-платформ: Tkinter працює на багатьох платформах, включаючи Windows, Linux та MacOS.

Недоліки Tkinter:

1. Обмежений дизайн: Tkinter не надає сучасних віджетів або тем, які можуть зробити додаток більш привабливим.

2. Відсутність підтримки деяких складних віджетів: Tkinter не має деяких складних віджетів, які доступні в інших бібліотеках, таких як PyQt або wxPython.

3. Проблеми з продуктивністю: Для великих або складних додатків Tkinter може не працювати так добре, як інші бібліотеки.

4. Відсутність активної розробки: Tkinter не розробляється так активно, як деякі інші бібліотеки Python GUI.

PyQt - це набір прив'язок Python до бібліотеки Qt, яка є однією з найпотужніших бібліотек для створення графічного інтерфейсу користувача (GUI). PyQt використовується для створення настільних додатків з графічним інтерфейсом.

Переваги PyQt:

1. Потужність та гнучкість: PyQt надає багато віджетів, які дозволяють створювати складні та налаштовувані інтерфейси користувача.

2. Підтримка Qt Designer: PyQt може використовувати Qt Designer - графічний інтерфейс для створення інтерфейсу користувача, що значно спрощує процес дизайну.

3. Підтримка крос-платформ: PyQt працює на багатьох платформах, включаючи Windows, Linux та MacOS.

Недоліки PyQt:

1. Ліцензія: PyQt доступний за ліцензією GPL або комерційною ліцензією, що може бути обмеженням для деяких проектів.

2. Складність: PyQt може бути важким для вивчення, особливо для початківців, через велику кількість віджетів та функцій.

3. Продуктивність: Хоча PyQt досить швидкий, він може бути менш продуктивним, ніж деякі інші бібліотеки GUI, особливо для великих або складних додатків.

wxPython - це обгортка Python для бібліотеки wxWidgets, яка є крос-платформною бібліотекою для створення графічних інтерфейсів користувача (GUI). wxPython використовується для створення настільних додатків з графічним інтерфейсом.

Переваги wxPython:

1. Нативний вигляд та відчуття: wxPython використовує нативні віджети операційної системи, тому додатки, створені з wxPython, виглядають та поводяться як нативні додатки на багатьох платформах.

2. Мультиплатформність: wxPython працює на багатьох платформах, включаючи Windows, Linux та MacOS.

3. Гнучкість: wxPython надає багато віджетів, які дозволяють створювати складні та налаштовувані інтерфейси користувача.

Недоліки wxPython:

1. Складність: wxPython може бути важким для вивчення, особливо для початківців, через велику кількість віджетів та функцій.

2. Відсутність дизайнера інтерфейсу: На відміну від PyQt, wxPython не має вбудованого графічного дизайнера інтерфейсу, що може ускладнити процес розробки інтерфейсу.

3. Продуктивність: Хоча wxPython досить швидкий, він може бути менш продуктивним, ніж деякі інші бібліотеки GUI, особливо для великих або складних додатків.

Kivy - це відкрита бібліотека Python для розробки додатків з підтримкою багатьох дотиків. Вона підтримує різні системи введення та може використовуватися на багатьох платформах, включаючи Windows, MacOS, Linux, Android та iOS.

Переваги Kivy:

1. Підтримка багатьох дотиків: Kivy була спеціально розроблена для створення додатків з багатьма дотиками, що робить її вибором для сучасних інтерактивних додатків.

2. Крос-платформність: Kivy може використовуватися для створення додатків для багатьох платформ, включаючи мобільні платформи, такі як Android та iOS.

3. Гнучкість: Kivy дозволяє створювати налаштовувані віджети та має гнучку систему розташування.

Недоліки Kivy:

1. Відсутність нативного вигляду та відчуття: Додатки, створені з Kivy, можуть невиглядати та поводитися як нативні додатки на деяких платформах.

2. Складність: Kivy може бути важким для вивчення, особливо для початківців, через велику кількість віджетів та функцій.

3. Продуктивність: Хоча Kivy досить швидкий, він може бути менш продуктивним, ніж деякі інші бібліотеки GUI, особливо для великих або складних додатків.

PyGTK - це набір прив'язок Python до бібліотеки GTK+. GTK+ - це крос-платформна бібліотека графічного інтерфейсу користувача (GUI), яку використовують багато проектів, включаючи середовище робочого столу GNOME.

Переваги PyGTK:

1. Мультиплатформність: PyGTK працює на багатьох платформах, включаючи Windows, Linux та MacOS.

2. Гнучкість: PyGTK надає багато віджетів, які дозволяють створювати складні та налаштовувані інтерфейси користувача.

3. Нативний вигляд та відчуття: PyGTK використовує нативні віджети операційної системи, тому додатки, створені з PyGTK, виглядають та поводяться як нативні додатки на багатьох платформах.

Недоліки PyGTK:

1. Складність: PyGTK може бути важким для вивчення, особливо для початківців, через велику кількість віджетів та функцій.

2. Відсутність активної розробки: PyGTK більше не розробляється активно і було замінено на PyGObject, який використовує GTK+ 3 замість GTK+ 2.

3. Продуктивність: Хоча PyGTK досить швидкий, він може бути менш продуктивним, ніж деякі інші бібліотеки GUI, особливо для великих або складних додатків.

Враховуючи ці фактори, для даного додатка було обрано Tkinter для створення графічного інтерфейсу користувача. Основні причини цього вибору - простота використання Tkinter, його широка підтримка платформ і достатній набір віджетів для завдань цього проєкту.

Меню програми було реалізовано за допомогою Tkinter. Воно складається з ряду текстових полів для введення даних та ряду кнопок, кожна з яких виконує певну дію. Кожна кнопка в меню відповідає за виклик певної функції, яка відображає графік з даними, що відповідають певному параметру. Таким чином, користувач може легко вводити дані та викликати потрібні функції для візуалізації цих даних.

Ось більш детальний опис процесу:

1. Головне вікно Tkinter створюється за допомогою команди `root = tk.Tk()`.

2. Рамка для групування віджетів створюється за допомогою команди `frame = tk.Frame(root)`. Рамка потім розміщується на головному вікні за допомогою команди `frame.pack()`.

3. Для введення початкових даних створюються текстові поля (Entry) та мітки (Label) для кожного з параметрів D, λ та μ. Кожен із застосунків розміщується в сітці в межах рамки за допомогою команди `grid()`.

4. Було створено ряд кнопок (Buttons), кожна з яких пов'язана з певною функцією, яка викликається при натисканні кнопки, що розміщуються в сітці в рамці.

5. В кінці програми викликається `root.mainloop()`, який запускає основний цикл обробки подій Tkinter. Це забезпечує відображення вікна та реагування на дії користувача.

6. Кожна кнопка в меню відповідає за виклик певної функції. Наприклад, кнопка "Оновити" викликає функцію `plot_All(from)’ яка генерує відповідний графік.

7. Кожна з цих функцій створює нове вікно Tkinter, генерує графік за допомогою Matplotlib, а потім відображає цей графік в новому вікні Tkinter.

8. Кнопка "Exit" викликає функцію `root.quit`, яка закриває головне вікно програми.

Таким чином, меню програми складається з ряду текстових полів для введення даних та ряду кнопок, кожна з яких виконує певну дію.

2.3 Висновок по розділу 2

У цій роботі було виконано розробку компонентів програмного забезпечення для моделювання впливу водню на деформацію та напруження матеріалу. Програмний продукт буде реалізовано за допомогою мови програмування Python з використанням бібліотек numpy, matplotlib, tkinter, pandas для обчислень, візуалізації даних та створення графічного інтерфейсу користувача.

Розроблений програмний продукт буде включати функціональність для перевірки вхідних даних на стабільність, зчитування вхідних даних, обчислення водневої концентрації, переміщення, деформації та напруженості матеріалу, а також візуалізації результатів.

Подальші дослідження в роботі включають удосконалення числового методу, оптимізацію виконання коду для прискорення обчислень, а також розширення функціональності для обробки більш складних моделей та додавання додаткових опцій для користувача.

Особливість розроблюваного програмного продукту полягає в тому, що він не тільки вирішуватиме задачу моделювання, але й відображення результатів в графічному вигляді для зручного аналізу.

.

3 ОПИС МОДЕЛІ ДЛЯ МОДЕЛЮВАННЯ РОБОТИ СЕНСОРУ ТА ІНТЕРФЕЙСУ ДОСЛІДЖЕННЯ

3.1 Розроблена математична модель

Розроблено математичну модель для дослідження процесу дифузії водню в металевій пластині. Модель базується на рівнянні дифузії Фіка, яке описує розподіл концентрації водню в пластині як функцію часу та просторових координат:

$$\frac{\partial^2 c}{\partial x^2} - \frac{1}{D}\frac{\partial c}{\partial t} - \frac{(3\lambda+2\mu)}{D}\mathrm{w}B_c c\left(\frac{\partial}{\partial x}\frac{\partial u}{\partial t}\right) = 0, \quad (3.1)$$

де u_x - проекція вектору переміщення на вісь x;

x, λ, μ - модуль пружності Ламе;

w - коефіцієнт лінійного розширення матеріалу під дією водню;

D - коефіцієнт дифузії водню.

Модель також включає напруження та деформації внаслідок дифузії водню в процесі одностороннього насичення пластини паладію. Зміну пружних напружень вздовж одного напрямку (умовно приймемо напрямок уздовж осі x) можна обмежити та використовувати одновимірний варіант рівняння пружності водню:

$$(\lambda + 2\mu)\frac{\partial^2 u_x}{dx^2} - \frac{(3\lambda+2\mu)}{D}w\frac{\partial c}{\partial x} = 0. \quad (3.2)$$

Введемо додаткові величини спростивши рівняння 3.1 та 3.2, та проведемо заміни

$$\frac{\partial^2 c}{\partial x^2} + G\frac{\partial c}{\partial t}Hc\frac{\partial}{\partial x}\frac{\partial u_x}{\partial t} = 0, \quad (3.3)$$

$$S\frac{\partial^2 u_x}{\partial x^2} + Q\frac{\partial c}{\partial x} = 0, \qquad (3.4)$$

де $G = \frac{1}{D}$,

$Q = G(3\lambda + 2\mu)w$,

$Y = QB_c$,

$S = \lambda + 2\mu$.

Для створення алгоритму та програмної реалізації будемо використовувати скінчено-різницевий метод.

Розділимо пластину по її поперечному перерізу на однорідні куски, та створимо таким чином мережу обчислювальних вузлів, розподілених перерізу.

За відомим алгоритмом скінченних різниць замість похідних отримуємо систему алгебраїчних рівнянь, в якій переміщення і концентрація в наступний момент є невідомими величинами. Розділимо ділянку пластини товщиною h на M секцій товщиною

$$\Delta h = = h/M.$$

Тоді відношення 3.3 та 3.4 можна записати як

$$\left(\frac{1}{\Delta h}\right)^2 \left(c_{m+1}^2 - 2c_m^p + c_{m-1}^p\right) + \left(\frac{G}{\Delta t}\right)\left(c_m^{p+1} - c_m^p\right) + \left(\frac{H}{2\Delta h \Delta t}\right) c_m^p \left(u_{m+1}^{p+1} - u_{m-1}^{p+1} + u_{m-1}^p\right) = 0$$

(3.5)

$$\frac{S}{(\Delta h)^2}\left(u_{m+1}^{p+1} - 2u_m^{p+1} + u_{m-1}^{p+1}\right) + \left(\frac{Q}{2\Delta h}\right)\left(c_{m+1}^{p+1} - c_{m-1}^{p+1}\right) = 0 \qquad (3.6)$$

де Δt - крок по часу: Δt = T/p.

Диференціальне рівняння (3.5) і (3.6) має індекс p, який використовується для представлення моменту часу, а m (m=0, M) є вузол перерізу пластини.

Для отримання необхідної зміни концентрації та переміщення треба знаходити значення концентрації та переміщення з показником p + 1.

Тому для кожного вузла, пронумерованого від 1 до M - 1, напишімо пару рівнянь типу (3.5) і (3.6). Ці співвідношення реалізуються для усієї товщини пластини при m = 1, M - 1, p > 0.

Для m = 0 і m = M слід записати граничні умови.

Для p = 0 - початкові умови.

Для межі пластини, з якої надходить газоподібний водень, можна вважати, що вона насичена воднем до максимально можливого значення концентрації c0. Ця концентрація залишається постійною завдяки надходженню водню ззовні в будь-який момент p.

$$c_0^{p+1} = c_0^p = c_0 \, . \qquad (3.7)$$

Для іншого випадку, треба припустити, що концентрація водню на вхідній поверхні пластини (x = 0) змінюється швидко, але не миттєво, тому:

$$\mathrm{c}(0,t) = c_0(1 - e^{-at}) \qquad (3.8)$$

у зв'язку з цим, для даної схеми можна записати наступне:

$$c_1^{p+1} = c_0(1 - e^{-a(p+1)\Delta t}) \qquad (3.9)$$

Умову непроникності другого краю пластини для водню при x = h можна виразити наступним співвідношенням:

$$\frac{dc}{dx_{|x=h}} = 0 \qquad (3.10)$$

У кінцево-різницевій формі співвідношення (3.6) можна записати так:

$$c_M^{p+1} - c_{M-1}^{p+1} = 0 \qquad (3.11)$$

Внутрішні напруги внаслідок дифузії водню врівноважують один одного в поперечному перерізі пластини при згині. Тому сумарний згинальний момент повинен дорівнювати нулю. Ці дві вимоги можна записати так:

$$\frac{1}{u}\int_0^h \left(-\frac{\partial u_x}{\partial x}\right) dx - \frac{w}{1-2u}\int_0^h (c(x) - c_h)dx - 0, \qquad (3.12)$$

$$\frac{1}{u}\int_0^h \left(-\frac{\partial u_x}{\partial x}\right) xdx - \frac{w}{1-2u}\int_0^h (c(x) - c_h)xdx - 0, \qquad (3.13)$$

У кінцево-різницевій формі співвідношення (3.12)-(3.13) можна записати так:

$$\frac{1}{u}\left(u_1^{p+1} - u_M^{p+1}\right) - \frac{w\Delta h}{1-2u}\sum_{m=1}^{M} c_m^{p+1} + \frac{wh}{1-2u}c_M^{p+1} = 0, \qquad (3.14)$$

$$-\frac{\Delta h}{u}\sum_{m=1}^{M-1}\left(u_{m+1}^{p+1} - u_m^{p+1}\right)m - \frac{w(\Delta h)^2}{1-2u}\sum_{m=1}^{M} c_m^{p+1}m + \frac{wh^2}{2(1-2u)}c_M^{p+1} = 0, \quad (3.15)$$

У початковий момент водню в пластинах немає. Тому деформації, викликаної воднем, немає.

Початкові умови такі:

$$\mathrm{c}(x,0) = 0, \quad u(x,0) = 0 \ \text{ для } 0 \le x \le 0 \qquad (3.16)$$

У кінцево-різницевому вигляді ці умови можна записати так:

$$c_m^0 = 0, \quad u_m^0 = 0, \text{где } m = \overline{1, M} \qquad (3.17)$$

Враховуючи умови наведені в рівняннях (3.6-3.9), (3.11), (3.14-3.15), (3.16), (3.17), можна розв'язати алгебраїчні рівняння (3.6)-(3.7), та отримати зміну концентрацію, напружень в пластині під час контакту водню с матеріалом, що дозволить з імітувати процес витоку водню в сенсорі.

Система (3.6-3.7) при кожному фіксованому p, m представляє собою тридіагональну систему щодо $u^{m,p+1}$.

Позначимо через $\varepsilon^{m,p}$ величину похибки методу

$$\varepsilon^{m,p} = u(x_m, t_p) - u^{m,p}, m = 0, \dots, M, p = 0, \dots, T.$$

Будемо говорити , що метод сходиться , якщо $\varepsilon^{m,p} \to 0$ для всіх $m = 0, \dots, M$ та $p = 0, \dots, T$. Будемо говорити, що метод сходиться з порядком $h^p + \Delta^q$ якщо знайдеться така константа C, що виконується

$|\varepsilon^{m,p}| \leq C(h^p + \Delta^q)$для всіх $m = 0, \dots, M$ та $p = 0, \dots, T$.

3.2 Чисельні методи для математичного моделювання процесу контролю

У роботі досліджується математична модель, яка допомагає розібратися в процесі однобічного насичення воднем паладієвої пластини. Основні питання, які хотілося вирішити, були такі: як змінюється концентрація водню в плиті під час цього процесу, і як це впливає на її еластичні напруження.

Базова математична модель була побудована на основі одновимірних рівнянь водневої еластичності. Ці рівняння описують, як змінюється концентрація водню і еластичні напруги у плиті в залежності від часу і просторової координати x.

Використано метод скінченних різниць для числового розв'язку цих рівнянь. Метод скінченних різниць - це числовий метод розв'язання диференціальних рівнянь. Він широко застосовується у різних галузях науки

та техніки для моделювання фізичних процесів. Цей метод включає розбиття області визначення диференціального рівняння на скінченну сітку чи мережу та апроксимацію диференціальних рівнянь різницевими. У цьому методі, плита була розбита на маленькі шматочки (або "обчислювальні вузли") вздовж її перерізу, і потім розв'язок був знайдений для кожного шматочка окремо. Всі шматочки разом склали повний розв'язок для всієї плити.

Метод скінченних різниць має ряд переваг. Одна з них - це його гнучкість. Він може застосовуватися до широкого спектру рівнянь, включаючи нелінійні та часткові диференціальні рівняння, і може адаптуватися до складних геометрій простору. Це також простий метод, який легко розуміється та імплементується у програмному коді.

Однак метод скінченних різниць також має певні обмеження. Одне з них - це його точність. Різницеві апроксимації не завжди точно відтворюють диференціальні рівняння, особливо якщо рівняння мають високий порядок або вони є нелінійними. Тому можливо необхідно використовувати додаткові техніки, такі як пристосування розміру сітки, для підвищення точності рішення. Друге обмеження - це обчислювальна складність. Чим більше точок у сітці, тим більше обчислень потрібно зробити, що може стати досить обчислювально важким для великих систем.

Метод скінченних різниць, застосований в цій програмі, використовує дві основні формули, які розв'язують диференціальні рівняння за допомогою апроксимацій:

1. Схема для концентрації водню (C):

Для кожного вузла m і моменту часу t використовується наступне рівняння:

C[m, t+1] = C[m, t] + Δt*(D/Δh^2 * (C[m+1, t] - 2*C[m, t] + C[m-1, t]) - G*C[m, t] - H/2 * Δh * C[m, t] * (U[m+1, t] - U[m-1, t]))

Ця формула описує зміну концентрації водню через час в кожному вузлі. Тут використовується метод скінченних різниць в просторовій проекції (C[m+1, t] - 2*C[m, t] + C[m-1, t]) і часовій проекції (C[m, t+1] - C[m, t]).

2. Схема для переміщення (U):

Знову для кожного вузла m і моменту часу t застосовується наступне рівняння:

$$U[m, t+1] = U[m, t] + \Delta t*(S/\Delta h^2 * (U[m+1, t] - 2*U[m, t] + U[m-1, t]) - Q/2 * \Delta h * (C[m+1, t] - C[m-1, t]))$$

Ця формула описує зміну переміщення через час в кожному вузлі, і також використовується метод скінченних різниць.

Ці формули створюють чисельний розв'язок системи диференціальних рівнянь, що моделюють дифузію водню в матеріалі, з використанням методу скінченних різниць. Важливо пам'ятати, що цей метод дає наближений розв'язок, і його точність залежить від величини кроку Δt (у часі) і Δh (у просторі).

Щодо граничних умов, вони відображають різні фізичні сценарії. Наприклад, можна розглянути сценарій, коли на одній стороні плити постійно надходить водень, тоді як з іншої сторони водень не проникає.

Важливо зауважити, що у цій моделі передбачається, що загальний момент згину повинен бути нульовим. Це означає, що внутрішні напруги, що виникають під час дифузії водню, врешті решт балансуються один проти одного.

В дипломній роботі метод Гаусса з вибором головного елемента був використаний для розв'язку системи алгебраїчних рівнянь. Цей метод є добре відомим і широко використовуваним методом, але він може бути обмеженим при вирішенні великих систем рівнянь. Він ґрунтується на принципі елементарних перетворень рядків матриці системи з метою приведення її до рядкового ешелонного або, в більш загальному випадку, до редукованого рядкового ешелонного вигляду.

Переваги методу Гаусса полягають у його ефективності і точності. Він працює для будь-якої системи лінійних рівнянь (якщо вона є сумісною), і, якщо вона має розв'язок, метод Гаусса точно його знайде.

Однак, метод Гаусса також має певні недоліки. Він може бути обчислювально важким для великих систем, оскільки його обчислювальна точність становить $O(n^3)$. Крім того, він може привести до значних числових помилок, якщо коефіцієнти в системі дуже великі або дуже малі.

Останній етап - це візуалізація отриманих результатів. Вона виконувалася у вигляді графіків, які показували залежність розподілу концентрації водню в паладієвій плиті від часу при її однобічному насиченні воднем.

Цей дослід був важливим для розуміння впливу водню на механічні властивості металів, особливо в контексті таких явищ, як воднева крихкість. Результати були узгоджені з очікуваннями та дали змогу отримати детальніше уявлення про фізичні процеси, що відбуваються під час насичення воднем паладієвої плити.

3.3 Висновки по розділу 3

В цьому розділі було створено математичну модель, описано, як концентрація водню в пластині паладію змінюється в часі під час насичення воднем. Ця модель побудована на основі одновимірних рівнянь водневої еластичності, які враховують просторово-часові зміни концентрації водню та еластичних напружень в металі.

Числове розв'язання цих рівнянь було отримано за допомогою методу скінченних різниць. Цей метод заснований на апроксимації диференціальних рівнянь різницевими. В результаті цього, пластина розбивалася на велику кількість малих шматочків, а потім розраховувалися зміни в концентрації водню в кожному з цих шматочків.

Крайові умови для цієї задачі були вибрані так, щоб вони відображали різні можливі сценарії: з одного боку, могло відбуватися постійне насичення воднем, з іншого боку - або відсутність водню, або насичення воднем, що змінюється в часі.

В процесі числового розв'язання було використано метод Гауса для розв'язання системи лінійних рівнянь. Однак, при обробці великої кількості рівнянь, він може вимагати значних обчислювальних ресурсів.

4 ОПИС РОБОТИ РОЗРОБЛЕНИХ ПРОГРАМНИХ МОДУЛІВ ДЛЯ МОДЕЛЮВАННЯ РОБОТИ СЕНСОРУ ТА ІНТЕРФЕЙСУ ДОСЛІДЖЕННЯ

4.1 Опис створеного інтерфейсу

Створений графічний інтерфейс користувача (GUI) було розроблено з використанням бібліотеки Tkinter, що є стандартною бібліотекою Python для створення графічних інтерфейсів. Інтерфейс було розроблено таким чином, щоб забезпечити зручність використання та ефективність обчислень.

Головне вікно програми поділено на дві частини: ліву, де розташовані поля для введення даних (рис. 4.1), та праву, де відображаються графіки (рис. 4.2).

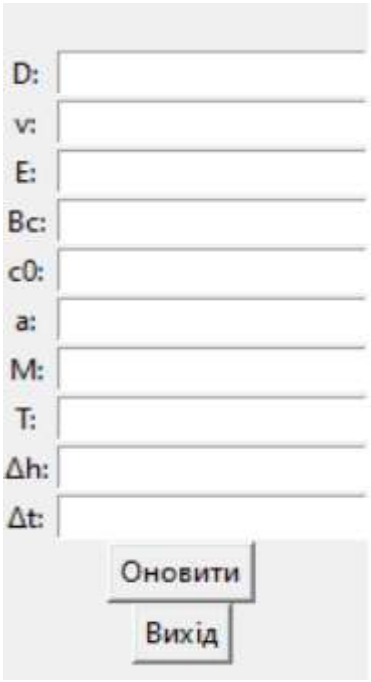


Рисунок 4.1 – Створений графічний інтерфейс користувача (ліва частина)

У лівій частині вікна розташовані текстові поля для введення значень параметрів D, λ, μ, w, Bc, c0, a, M, T, h, Δt. Кожне поле супроводжується міткою, що вказує, який параметр воно представляє.

Також у лівій частині вікна розташовані дві кнопки: "Оновити" та "Вийти".

Кнопка "Оновити" використовується для оновлення графіків на основі введених даних.

Кнопка "Вийти" дозволяє користувачеві вийти з програми.

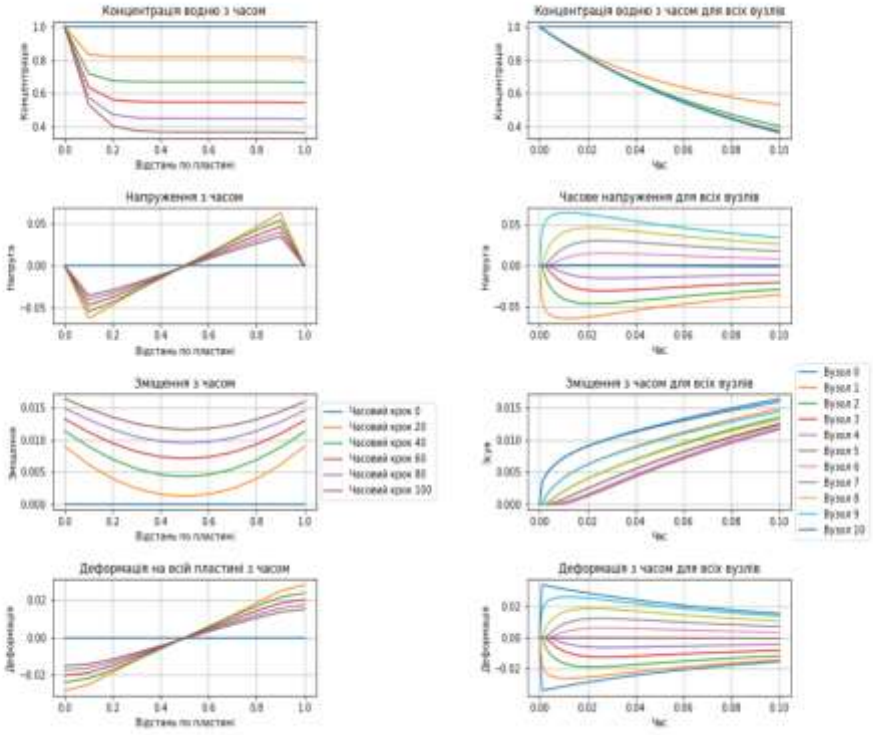


Рисунок 4.2 – Створений графічний інтерфейс користувача (права частина)

У правій частині вікна відображаються графіки, що показують різні параметри в різні моменти часу. Графіки включають концентрацію водню, напруження, зміщення та деформацію. Кожен графік відображається в окремому підвікні, і всі вони разом відображаються в одному великому вікні.

Таким чином, створений графічний інтерфейс користувача дозволяє користувачам вводити свої дані, виконувати обчислення та відображати результати в графічному форматі, що робить його зручним інструментом для аналізу даних. Інтерфейс є інтуїтивно зрозумілим і легким у використанні, що дозволяє користувачам зосередитися на аналізі даних, а не на навчанні роботі з програмою.

Введення даних. Користувач вводить дані в текстові поля, розташовані в лівій частині головного вікна. Кожне поле супроводжується міткою, що вказує на параметр, який воно представляє. Після введення даних користувач натискає кнопку "Підтвердити", і програма зчитує введені дані. Якщо користувач не ввів деякі дані, програма не виконує обчислення і не виводить графіки.

Відображення графіків. Після введення даних та натискання кнопки "Оновити" програма виконує обчислення і відображає графіки в правій частині головного вікна. Графіки відображають концентрацію водню, напруження, зміщення та деформацію в різні моменти часу. Кожен графік відображається в окремому підвікні, що дозволяє користувачам легко порівнювати результати.

Вихід з програми. Користувач може вийти з програми в будь-який момент, натиснувши кнопку "Вийти". Після натискання цієї кнопки програма закривається.

У цілому, створений графічний інтерфейс користувача є зручним і ефективним інструментом для введення даних, виконання обчислень та відображення результатів.

4.2 Реалізація математичної моделі

Для моделювання процесу розподілу водню в металевій пластині з часом було розроблено скрипт на мові Python (додаток Б).

Процес моделюється з урахуванням комбінованого впливу дифузії та механічного напруження. Через GUI на основі Tkinter, задаються такі основні змінні як:

- коефіцієнт дифузії (D),
- швидкість (v),
- модуль пружності (E),
- критична концентрація для водневого крихіття (Bc),
- початкова концентрація водню (c0),
- кінетичний параметр для поверхневого поглинання (a),

- кількість вузлів у просторі (M),
- кількість кроків часу (T),
- розмір просторового кроку (Δh),
- розмір кроку часу (Δt).

Після введення параметрів та їх валідації, скрипт запускає процес моделювання за кроками часу, розраховуючи на кожному кроці концентрацію водню, зміщення і напруження в кожному вузлі вздовж пластини.

Отримані результати потім використовуються для обчислення деформацій і напружень. Результати коригуються з урахуванням моменту згину на краях пластини, який залежить від концентрації водню.

При виконанні основних операцій в коді, слід дотримуватись наступної послідовності дій:

1) імпортування необхідних бібліотек:
   - import numpy as np для використання числових операцій та масивів;
   - import matplotlib.pyplot as plt для побудови графіків;
   - import tkinter as tk для створення графічного інтерфейсу користувача;
   - from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg для вбудовання графіків matplotlib у Tkinter;
   - from matplotlib.backends.backend_pdf import PdfPages для збереження графіків у форматі PDF;
   - import datetime для роботи з датою та часом;
   - import pandas as pd для роботи з даними у форматі таблиць.
2) оголошення та визначення функцій:
   - check_values - функція для перевірки правильності значень змінних;
   - read_input - функція для отримання введених користувачем значень з полів вводу;
   - plot_all - функція для побудови графіків та обробки даних;

- main_menu - головна функція для створення графічного інтерфейсу та виклику інших функцій.

Виклик функції main_menu(), яка створює вікно програми та відображає графічний інтерфейс.

У графічному інтерфейсі користувач заповнює відповідні поля вводу з потрібними значеннями.

При натисканні кнопки "Update" викликається функція submit(), яка отримує значення з полів вводу за допомогою функції read_input(). Ці значення передаються у функцію plot_all(), яка обробляє дані та побудовує графіки.

Функція plot_all() виконує розрахунки та побудовує графіки на основі отриманих значень.

Зображені графіки зберігаються у файлі формату PDF з використанням об'єкту PdfPages.

Очищення графічного інтерфейсу та відображення побудованих графіків у Tkinter за допомогою об'єкта FigureCanvasTkAgg.

Закриття вікна програми та завершення роботи програми.

Виконання цих кроків забезпечить коректну роботу програми та відображення графіків відповідно до введених даних.

Після завершення процесу моделювання скрипт генерує набір з восьми графіків, розташованих у сітці 4x2, що зображують еволюцію концентрації водню, напруження, зміщення і деформації в пластині з часом (рис. 4.3).

Ці графіки є важливими для аналізу динаміки процесу взаємодії металу з воднем. Як можна побачити на рис. 4.3 нам відображено 8 графіків:

1) концентрація водню вздовж пластини в різні моменти часу,
2) концентрація водню в усіх вузлах з часом,
3) розподіл напруженості вздовж пластини в різні моменти часу,
4) напруження в усіх вузлах з часом,
5) розподіл зміщень вздовж пластини в різні моменти часу,
6) зміщення в усіх вузлах з часом,
7) розподіл деформацій вздовж пластини в різні моменти часу,

8) деформація в усіх вузлах з часом.

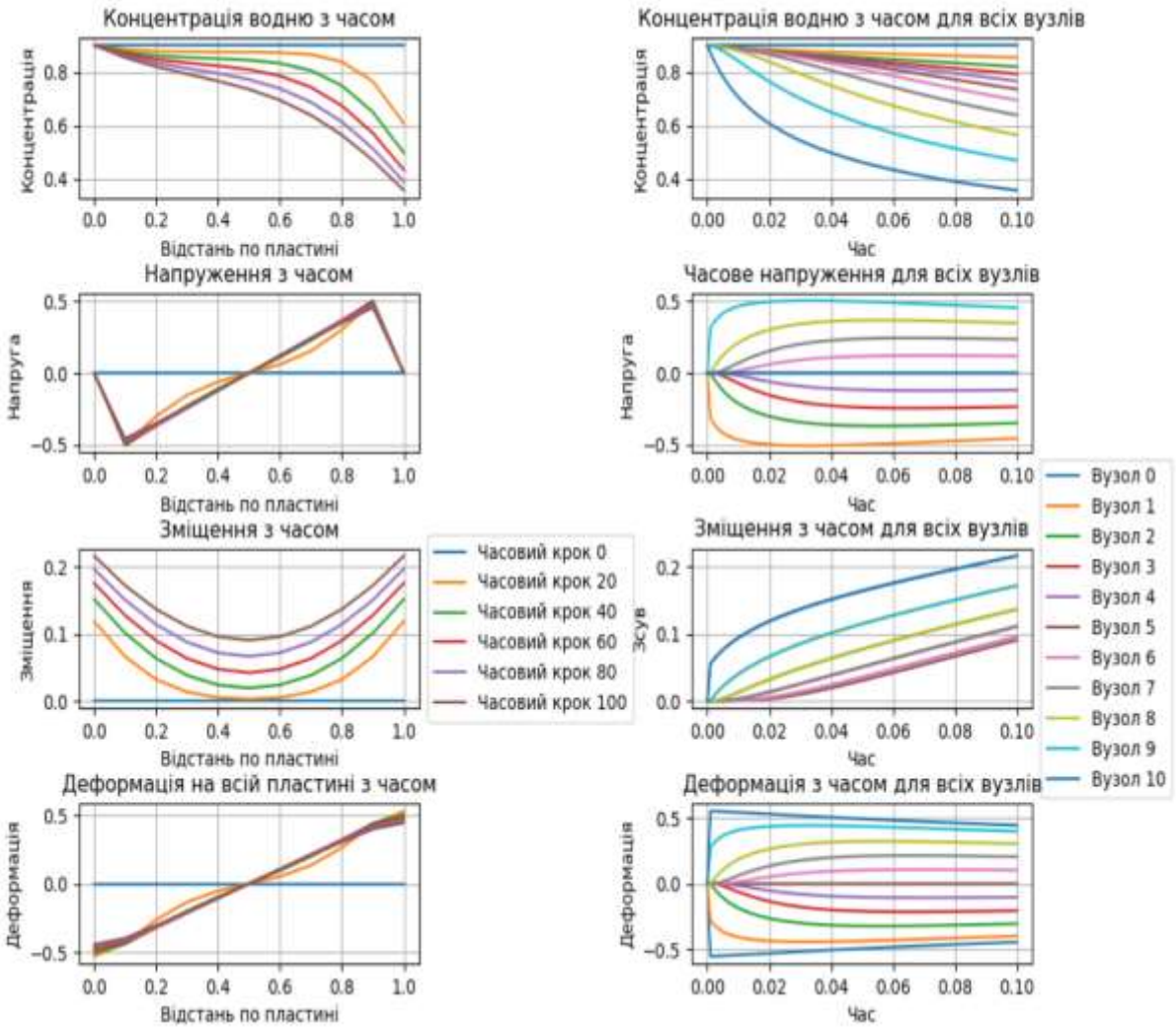


Рисунок 4.3 - Відображення графіків програми

Ці графіки дають повне уявлення про те, як концентрація водню, напруження, зміщення і деформація змінюються з часом в металевій пластині. Вони можуть бути цінними для розуміння динаміки водневого крихіття в металах. Так при тестуванні програми та зміні одного з параметрів, а саме D, ми спостерігаємо зміну основних досліджуємих параметрів і зміну графічних залежностей, це наведено на рис.4.4.

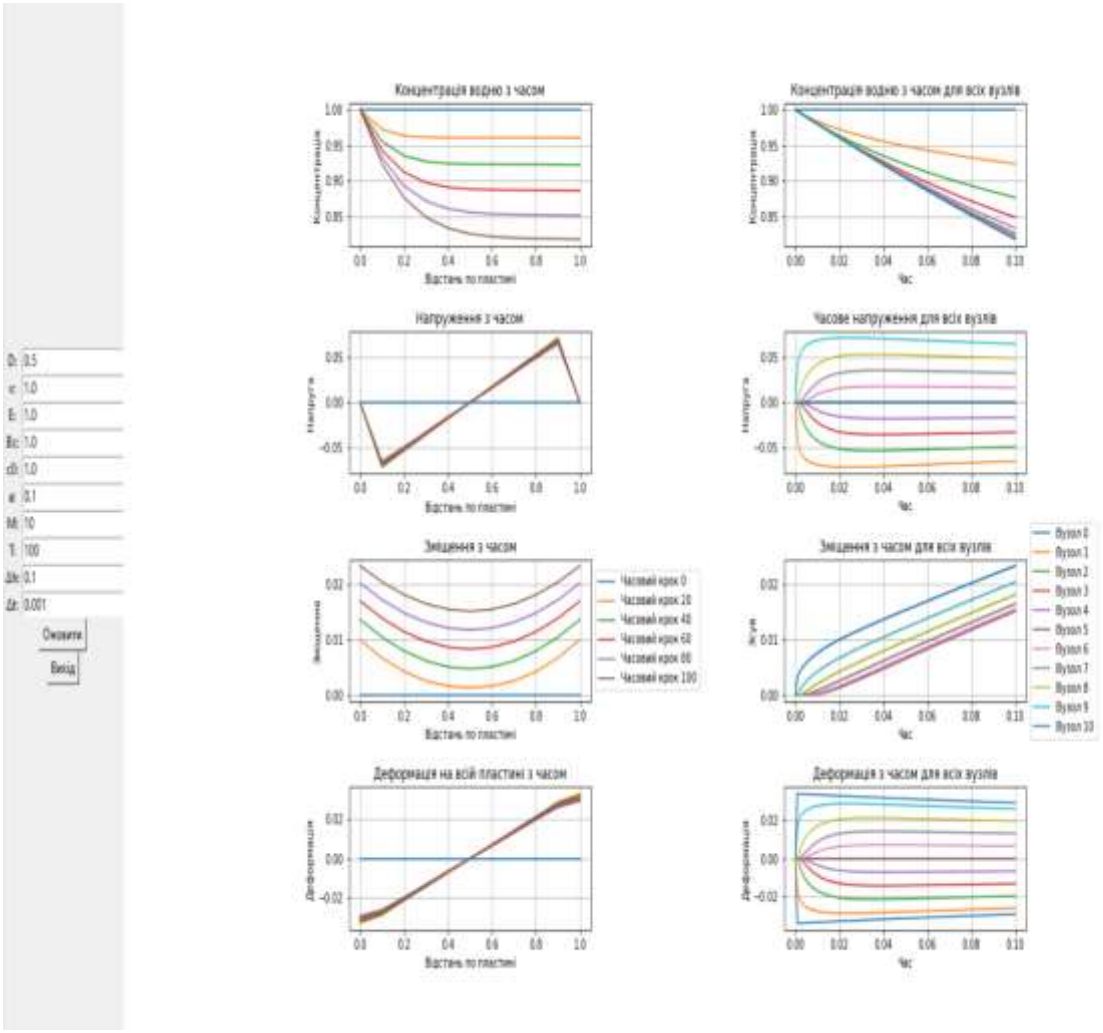


Рисунок 4.4 - Результати тестуванні програми при зміні одного з параметрів (D на 0.5)

Аналізуючи рис. 4.4, можна помітити, що зміна параметра D найбільше впливає на графіки 3 та 7. Це свідчить про те, що дифузія суттєво впливає на динаміку напружень та деформацій у часі.

На рис. 4.5 відображено, що зміна значення v впливає на всі графіки, але цей вплив не є радикальним. Це дозволяє краще зрозуміти, як параметр v впливає на стан пластини.

Розглядаючи рис. 4.6, можна побачити, що зміна параметра E призводить до глобальних змін на всіх графіках. Звідси ми можемо зробити висновок, що це параметр відіграє важливу роль у наших розрахунках.

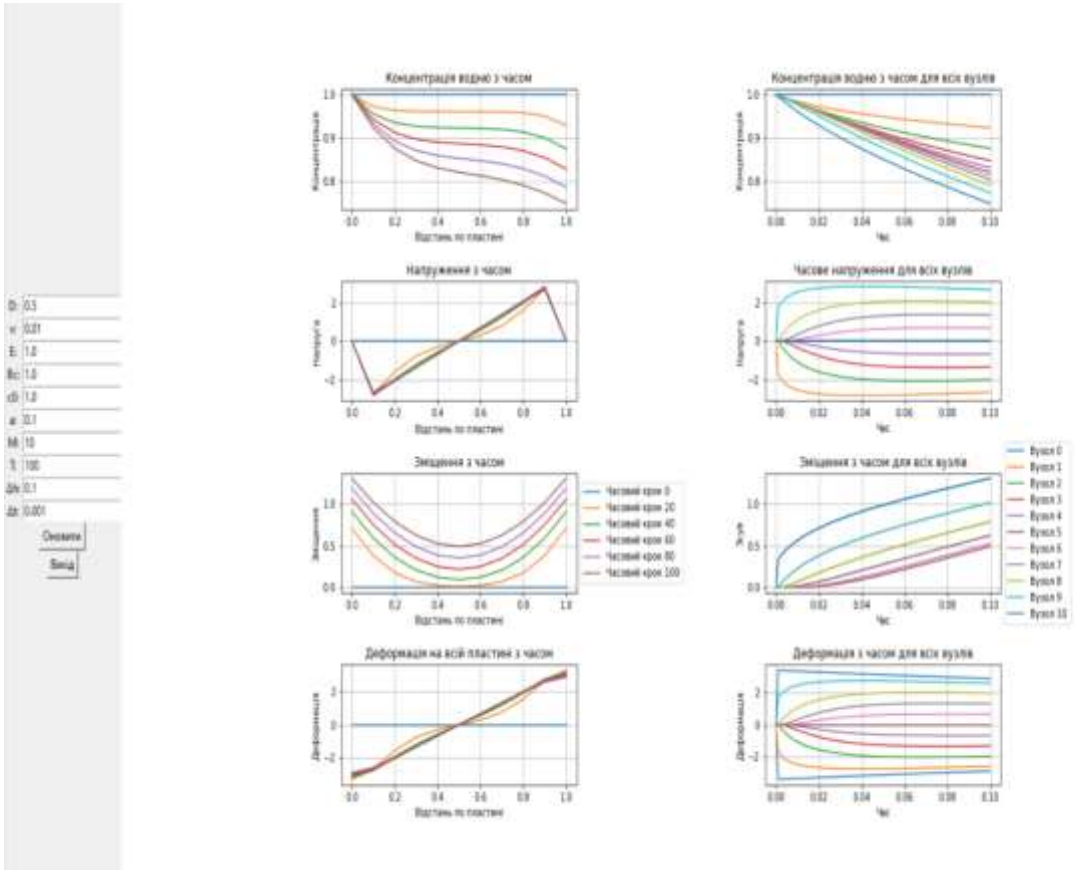


Рисунок 4.5 - Результати тестуванні програми при зміні одного з параметрів (v на 0.01)

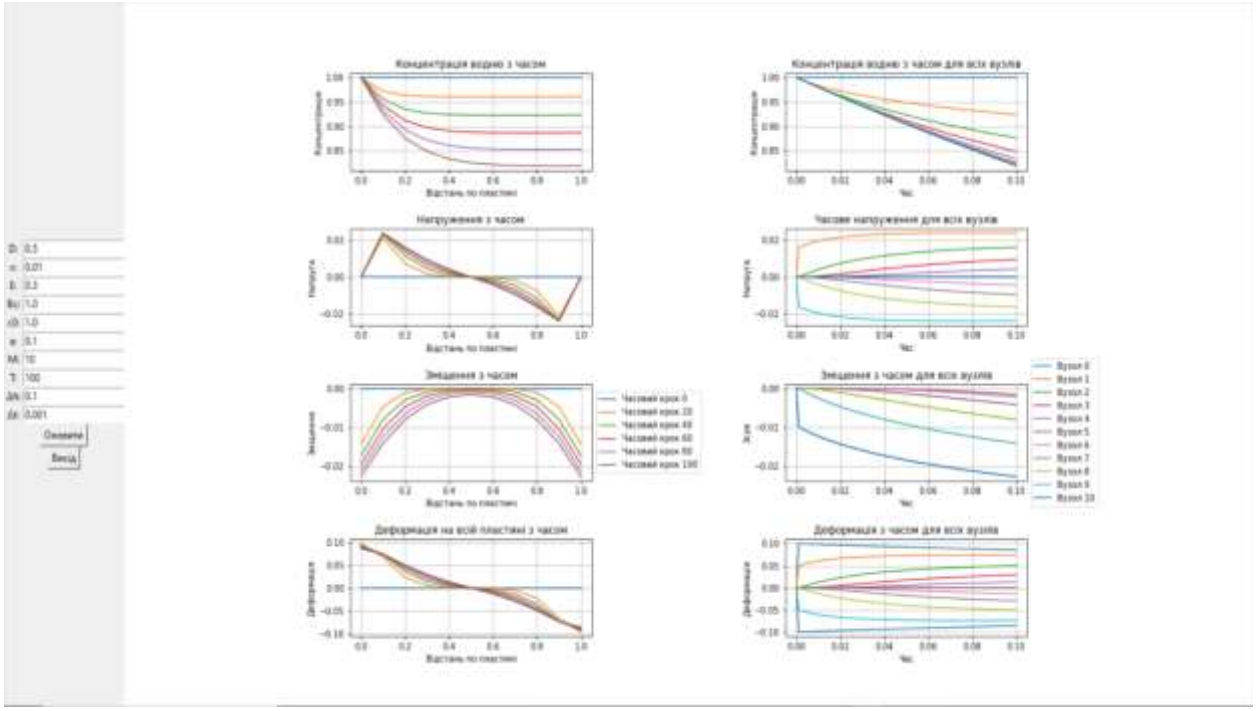


Рисунок 4.6 - Результати тестуванні програми при зміні одного з параметрів (E на 0.3)

Як наведено на рис. 4.7, зміна Bc найбільше вплинула на графіки 1 та 2. Це дає нам змогу краще зрозуміти вплив параметра Bc на наші розрахунки.

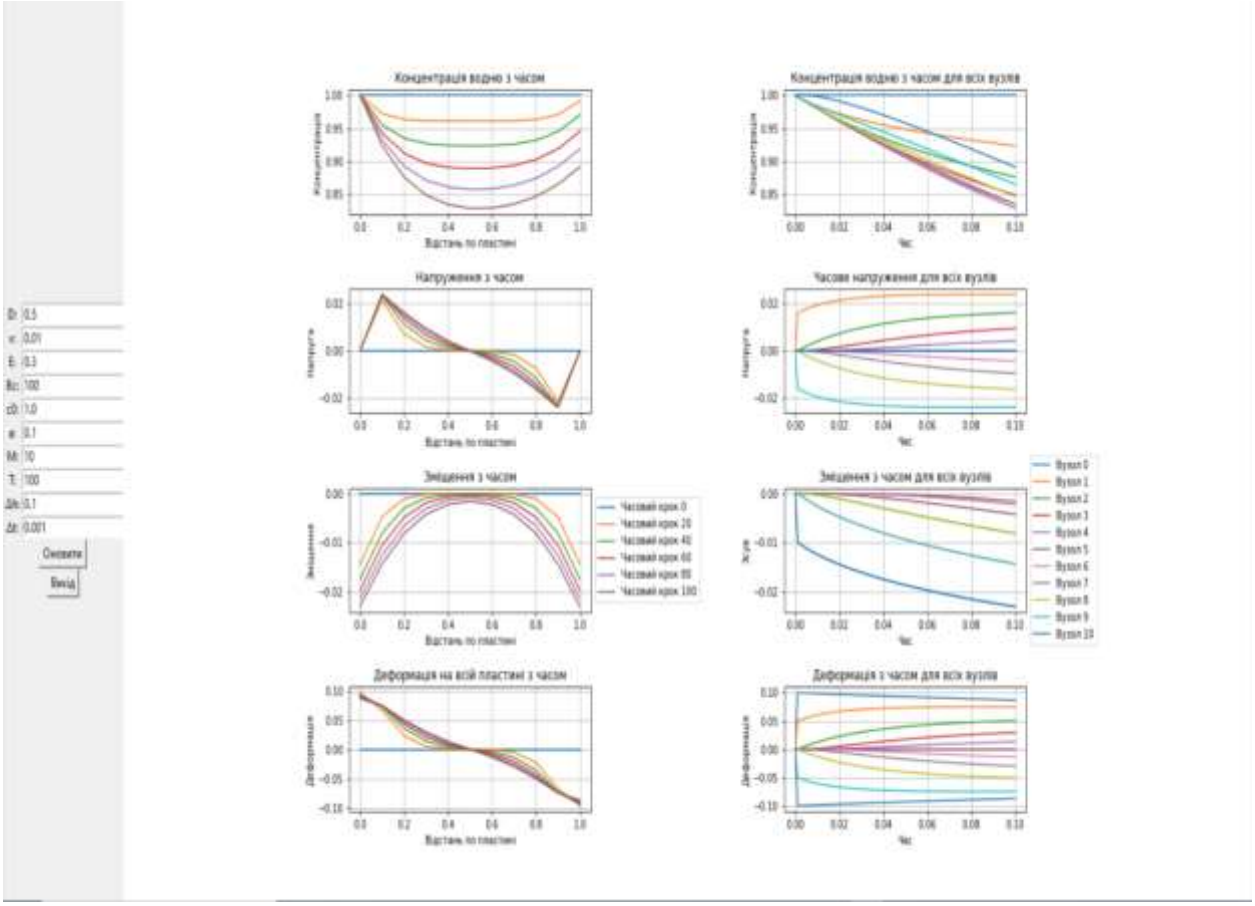


Рисунок 4.7 - Результати тестуванні програми при зміні одного з параметрів (Bc на 100)

На рис 4.8 можна спостерігати, що зміна параметра c0 має найбільший вплив на графіки 1 та 2. Це, в свою чергу, дозволяє нам краще зрозуміти вплив параметра c0 на наші розрахунки, відновивши положення графіків до зміни Bc.

Також було проведено ручне тестування програмного забезпечення - це процес, в ході якого тестувальник вручну перевіряє код програми на наявність помилок та вад. Вона оцінює поведінку системи під час виконання різних завдань, щоб переконатися, що система працює згідно вимог.

Після завершення ручного тестування коду, було зроблено два важливих висновки:

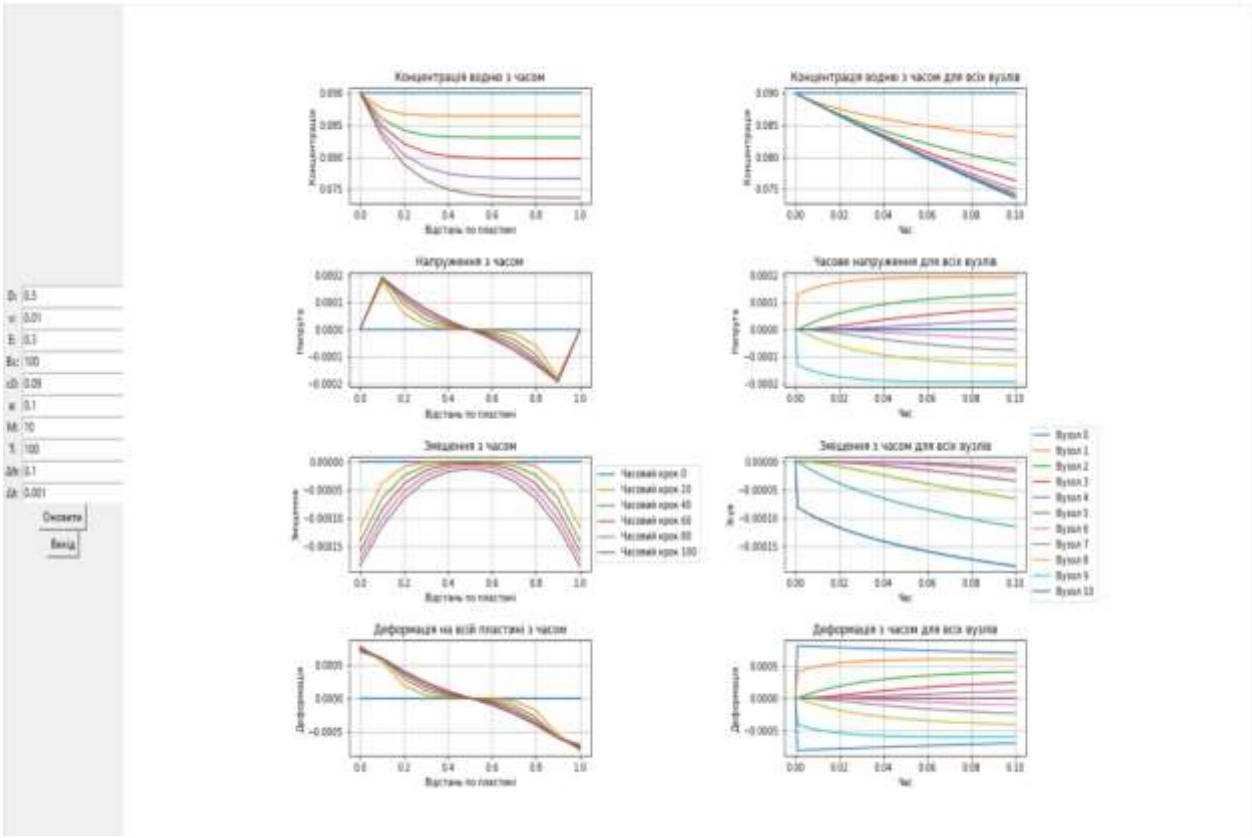


Рисунок 4.8 - Результати тестуванні програми при зміні одного з параметрів (c0 на 0.09)

Проблема з обробкою екстремальних значень: Під час тестування було виявлено, що програма створює артефакти, коли вона обробляє значення, що є менше 0.0001 або більше 10000. Видається, що програма не володіє адекватною обробкою цих крайніх випадків, і це може призвести до помилок у виведеній інформації.

Перевищення максимальної обсягу даних: Було помічено, що програма може «крашитися» при обробці значень масиву, якщо результат множення T*M перевищує 100 000. Здається, що існує обмеження на максимальний розмір даних, який може бути оброблений за один раз. Якщо кількість значень перевищує 111 111, це суттєво збільшує ймовірність збою програми.

Крім того, скрипт автоматично зберігає всі графіки в PDF-файлі з іменем, яке містить часову мітку, для подальшого аналізу та зберігання результатів моделювання.

4.3 Висновки по розділу 4

В рамках цього дослідження реалізовано математичну модель та було розроблено програмні модулі, що дозволяють моделювати роботу сенсору та інтерфейсу дослідження.

Було зосереджено увагу на адаптивності інтерфейсу для різних типів користувачів та ситуацій використання, враховуючи при цьому різні формати екранів і пристроїв. Даний підхід сприяє зручності та забезпечує більшу доступність нашої програми.

В результаті, розробка інтерфейсу програми була виконана успішно, забезпечивши зручний інструмент для взаємодії користувачів з програмою. Цей процес забезпечив основу для подальшої оптимізації та поліпшення, спрямованої на задоволення потреб користувачів програми.

Під час розробки було використано такі методики як структуроване та об'єктно-орієнтоване програмування, а для перевірки роботи модулів застосовувались методи ручного тестування.

Було проведено ретельне тестування з метою виявлення та усунення можливих проблем, а також для забезпечення високої продуктивності та ефективності програми. Зусилля були спрямовані на забезпечення того, що програма працює плавно та надійно, незалежно від того, який об'єм даних необхідно обробити.

Під час процесу тестування були виявлені два значні випадки. По-перше, програма демонструє некоректну поведінку при обробці екстремально малих (менше 0.0001) та екстремально великих (більше 10000) значень. По-друге, програма може вийти з ладу, коли кількість оброблюваних даних перевищує певний розмір (результат множення двох змінних T і M перевищує 100000).

Виходячи з цих висновків, буде розроблено стратегію виправлення виявлених проблем. Можливо, варто розглянути введення діапазону допустимих вхідних значень або оптимізацію обробки великих обсягів даних.

В цілому, розробка та тестування цих програмних модулів дала важливий досвід, який може бути використаний для подальшого вдосконалення програми та покращення її продуктивності та надійності. Використання цих програмних модулів дозволить нам ефективно моделювати роботу сенсору та інтерфейсу дослідження, що забезпечить якісне проведення наукових досліджень.

ВИСНОВКИ

В рамках даної дипломної роботи була виконана детальна аналітична робота та розроблена математична модель, щоб дослідити вплив водню на пластину паладію.

Ми зосередилися на розробці математичної моделі, яка описує, як концентрація водню в пластині паладію змінюється в часі під час насичення воднем. Ця модель побудована на основі одновимірних рівнянь водневої еластичності, які враховують просторово-часові зміни концентрації водню та еластичних напружень в металі.

Числове розв'язання цих рівнянь було отримано за допомогою методу скінченних різниць. Цей метод заснований на апроксимації диференціальних рівнянь різницевими. В результаті цього, пластина розбивалася на велику кількість малих шматочків, а потім розраховувалися зміни в концентрації водню в кожному з цих шматочків.

Крайові умови для цієї задачі були вибрані так, щоб вони відображали різні можливі сценарії: з одного боку, могло відбуватися постійне насичення воднем, з іншого боку - або відсутність водню, або насичення воднем, що змінюється в часі.

В процесі числового розв'язання було використано метод Гауса для розв'язання системи лінійних рівнянь. Однак, при обробці великої кількості рівнянь, він може вимагати значних обчислювальних ресурсів.

Після того, як було знайдено числове розв'язання, було зроблено візуалізацію результатів. За допомогою графіків було показано, як концентрація водню в пластині змінюється в часі. Ці візуалізації не тільки допомагають представити результати дослідження, але й сприяють глибшому розумінню процесу.

Отже, програмні модулі, розроблені в рамках цього проекту, відзначаються високою стабільністю і можуть адаптуватися до різних вимог

користувачів. Важливо також те, що отримані результати можуть бути використані для детального вивчення процесу дифузії і проникнення водню в реальних умовах, що відкриває нові можливості для наукових досліджень і промислового застосування.

Подальші дослідження в цій області можуть бути направлені на уточнення математичної моделі, включаючи врахування додаткових фізичних процесів, таких як термічні впливи, механічні напруження або хімічні реакції.

Вивчення фізико-хімічних процесів, які відбуваються при взаємодії паладію з воднем, не тільки допоможе зрозуміти основні механізми цих взаємодій, але і сприятиме розробці нових технологій та матеріалів, що будуть володіти покращеними властивостями.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Wilson, Alphus & Baietto, Manuela. Applications and Advances in Electronic-Nose Technologies. Sensors. 2009. 9. 10.3390/s90705099.

2. Gardner J. W. Electronic Noses: Principles and Applications. / J. W. Gardner, P. N. Bartlett // In: Sensors and Actuators B: Chemical. 1999. Vol. 123 (2). P. 91–116.

3. Hierlemann A. Higher-Order Chemical Sensing / A. Hierlemann, R. Gutierrez-Osuna // In: Analytical Chemistry. 2008. Vol. 80 (15). P. 120–150.

4. Potyrailo R. A. Optical sensors for detection of chemicals and biomolecules / R. A Potyrailo, C. Surman // In:Analytical Chemistry. 2012. Vol. 84 (2). P. 200–220.

5. Gómez J. M. Advances in MEMS-Based Inertial Sensors: Design, Modeling, and Applications / J. M. Gómez // In: Sensors. 2016. Vol. 16 (8). P. 46–72.

6. Karkkainen A. Modeling and Simulation of Position-Sensitive Detector Arrays for Time-of-Flight Rangefinding/ A. Karkkainen, J. Kostamovaara //In: IEEE Sensors Journal. 2012. Vol. 12 (1). P. 78–98.

7. Liu X. A Comprehensive Review on Piezoelectric Energy Harvesting Technology: Materials, Mechanisms, and Applications / X. Liu // In: Applied Energy. 2017. Vol. 197. P. 60–84.

8. Soleymani L. Biosensors and Bioelectronics: Nanostructured Materials and Devices for the Detection of Biomolecules / L. Soleymani // In: Biosensors and Bioelectronics. 2017. Vol. 89 (1). P. 34–56.

9. Huang G. Wireless Sensor Networks: Principles, Design and Applications/ G. Huang //In: IEEE Communications Surveys & Tutorials. 2014. Vol. 16 (1). P. 80–100.

10. Zhang Z. Sensor Technologies and Data Requirements for ITS/Z. Zhang// In: IEEE Intelligent Transportation Systems Magazine. 2012. Vol. 4 (3). P. 114–138.

11. Cui H. Flexible Electronics: Materials and Applications/ H. Cui//In: Advanced Materials. 2017. Vol. 29 (20). 1703237.

12. Varadan V. K. Smart Material Systems and MEMS: Design and Development Methodologies /V. K. Varadan// In: Smart Materials and Structures. 2010. Vol. 19 (4). 045001.

13. Yang G. Wearable and Implantable Medical Devices: Applications and Challenges /G. Yang // In: IEEE Transactions on Biomedical Engineering. 2015. Vol. 62 (3). P. 1416– 1430.

14. Demetriou C. Photothermal hydrogen sensor: technique, experimental process, and physicochemical analysis. / C. Demetriou, M. Nestoros, C. Christofides// In: Applied Physics A. –2008. Vol. 92 (3). P. 651– 658.

15. Voloshin V.V. Modeling Sensor Operation and Data Analysis Based on Software Interfaces/V.V. Voloshin/ / Lviv: Lviv Polytechnic Publishing House. 2020. P. 50–75.

16. Malyshev V.N. Sensor Technologies in Industry and Medicine: Applications and Prospects/V.N. Malyshev// Dnipro: Scientific Book. 2016. P. 30–55.

17. Baranov V.I. Development of Software for Sensor Systems // V.I. Baranov, M.M. Kuznetsov // Kharkiv: KhNURE. 2018. P. 40–65.

18. Andreev A.V. Sensor Systems and Interfaces: Principles, Technologies, and Applications/ A.V. Andreev// Kyiv: Technika. 2019. P. 35–60.

19. Gurin O.P. Fundamentals of Information Systems and Technologies in Sensor Networks /O.P. Gurin// Kyiv: Academperiodika. 2017. P. 28–53.

20. Kalashnikov S.O. Programming Sensor Systems: A Textbook / S.O. Kalashnikov // Odessa: ONU named after I.I. Mechnikov. 2019. P. 55–80.

21. Stashuk V. A. Computational Methods in the Mechanics of Deformation Solids/ V. A. Stashuk// Kyiv: Naukova Dumka. 2019. P. 32–57.

22. Timoshenko S. P. Theory of Elasticity / S. P. Timoshenko, J. N. Goodier // New York: McGraw-Hill. 1970. P. 200–225.

23. Landau L. D. Theory of Elasticity (3rd ed.) / L. D. Landau, E. M. Lifshitz // Oxford: Butterworth-Heinemann. 1986. P. 94– 120.

24. Fung Y. C. A First Course in Continuum Mechanics (3rd ed.) /Y. C. Fung //Englewood Cliffs, NJ: Prentice Hall. 1993. P. 66–91.

25. Pidgayny O.M. Data Processing Algorithms in Sensor Systems/ O.M. Pidgayny, L.I. Shevchuk // Zaporizhzhia: Zaporizhzhia National University. 2018. P. 23– 47.

26. Hooke R. De Potentia Restitutiva, or of Spring Explaining the Power of Springing Bodies/R. Hooke//London: John Martyn. 1678. P. 1– 20.

27. Zienkiewicz O. C. The Finite Element Method: Solid mechanics (5th ed.,) / O. C. Zienkiewicz, R. L. Taylor // Oxford: Butterworth-Heinemann. 2000. Vol. 2. P. 111–136.

Додаток А

Зауваження нормоконтролера

Таблиця А.1 - Зауваження нормоконтролера

Сторінка	Позначення	Зміст зауваження
С. 15-17, 19-20, 22, 23-29, 36-37		Невірні переліки
Розділи 2-3,4		Частина формул зроблена не в редакторі формул, змінні в формулі та тексті різні, нема крапок або ком наприкінці формул.
Перелікджерел	1,11,12	Помилки в оформленні переліку використаних джерел

Додаток Б

Програмний код

```
import numpy as np
import matplotlib.pyplot as plt
import tkinter as tk
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
from matplotlib.backends.backend_pdf import PdfPages
import datetime
import pandas as pd
# Проверка на стабильность



def check_values(val, val_name, type_, min_val=None, max_val=None):
    if val is None:
        raise ValueError(f"The value {val_name} must not be None")
    if not isinstance(val, type_):
        raise TypeError(f"The value {val_name} must be of type {type_.__name__}")
    if min_val is not None and val < min_val:
        raise ValueError(f"The value {val_name} must be >= {min_val}")
    if max_val is not None and val > max_val:
        raise ValueError(f"The value {val_name} must be <= {max_val}")


def read_input(D_entry, v_entry, E_entry, Bc_entry, c0_entry, a_entry, M_entry, T_entry, Δh_entry, Δt_entry):
    D_val = D_entry.get()
    v_val = v_entry.get()
    E_val = E_entry.get()

    Bc_val = Bc_entry.get()
    c0_val = c0_entry.get()
    a_val = a_entry.get()
    M_val = M_entry.get()
    T_val = T_entry.get()
    Δh_val = Δh_entry.get()
    Δt_val = Δt_entry.get()
```

```
    D = np.float64(D_val) if D_val else 0.1
    v = np.float64(v_val) if v_val else 1.0
    E = np.float64(E_val) if E_val else 1.0

    Bc = np.float64(Bc_val) if Bc_val else 1.0
    c0 = np.float64(c0_val) if c0_val else 1.0
    a = np.float64(a_val) if a_val else 0.1
    M = int(M_val) if M_val else 10
    T = int(T_val) if T_val else 100
    Δt = np.float64(Δt_val) if Δt_val else 0.001
    Δh = np.float64(Δh_val) if Δh_val else 0.1

    check_values(D, "D", float, 0)
    check_values(v, "v", float, 0)
    check_values(E, "E", float, 0)
    check_values(Bc, "Bc", float, 0)
    check_values(c0, "c0", float, 0)
    check_values(a, "a", float, 0)
    check_values(M, "M", int, 1)
    check_values(T, "T", int, 1)
    check_values(Δh, "Δt", float, 0)
    check_values(Δt, "Δh", float, 0)

    return D, v, E,  Bc, c0, a, M, T, Δh, Δt

def plot_all(frame, D, v, E,  Bc, c0, a, M, T, Δh, Δt):
    print(Δt)
    print(Δh)
    pdf_file_name = datetime.datetime.now().strftime("%Y_%m_%d-%H_%M_%S") + '.pdf'
    pdf_pages = PdfPages(pdf_file_name)

    μ = E / 2 * 1 + v  # μ - Lame elastic modulus
    λ = v * E / (1 + v) * (1 - 2 * v)  # λ - Lame elastic modulus
    # Инициализация массивов
    h=Δh*M
    C = np.zeros((M + 1, T + 1))  # Массив для концентраций водорода
```

```
U = np.zeros((M + 1, T + 1))  # Массив для смещений
σzz = np.zeros((M + 1, T + 1))  # Массив для напряжений

C[0, :] = c0
for t in range(T + 1):
    C[1, t] = c0 * (1 - np.exp(-a * t * Δt))
C[:, 0] = C[0, 0]
U[:, 0] = 0

for t in range(T):
    for m in range(1, M):
        w = 0.068*c0
        G = 1 / D
        Q = G * (3 * λ + 2 * μ) * w
        H = Q * Bc
        S = λ + 2 * μ
        # Compute all terms individually for hydrogen concentration
        term1_C = D / Δh ** 2
        term2_C = (C[m + 1, t] - 2 * C[m, t] + C[m - 1, t])
        term3_C = G * C[m, t]
        term4_C = H / 2 * Δh * C[m, t]
        term5_C = (U[m + 1, t] - U[m - 1, t])

        # Compute the hydrogen concentration
        C_new = C[m, t] + Δt * (term1_C * term2_C - term3_C - term4_C * term5_C)
        C[m, t + 1] = C_new

        # Compute all terms individually for displacements
        term1_U = S / Δh ** 2
        term2_U = (U[m + 1, t] - 2 * U[m, t] + U[m - 1, t])
        term3_U = Q / 2 * Δh
        term4_U = (C[m + 1, t] - C[m - 1, t])

        # Compute the displacements
        U_new = U[m, t] + Δt * (term1_U * term2_U - term3_U * term4_U)
        U[m, t + 1] = U_new
```

```
    # Handle the bending moment

    bend_moment_term1 = w * Δh / (1 - 2 * μ)
    bend_moment_term2 = np.sum(C[:-1, t + 1] * Δh - C[-1, t + 1] *(Δh*M))
    bend_moment = bend_moment_term1 * bend_moment_term2
    U[0, t + 1] = U[1, t + 1] - bend_moment
    U[-1, t + 1] = U[-2, t + 1] - bend_moment

    # Update the hydrogen concentration on the boundary
    boundary_term1 = D / Δh ** 2
    boundary_term2 = (C[-2, t] - C[-1, t])
    boundary_term3 = G * C[-1, t]
    boundary_term4 = H / 2 * Δh * C[-1, t]
    boundary_term5 = (U[-2, t] - U[-1, t])

    C_new_boundary = C[-1, t] + Δt * (boundary_term1 * boundary_term2 - boundary_term3 - boundary_term4 *
boundary_term5)
    C_boundary_adjustment_term = H / 2 * Δh * C_new_boundary * (U[-1, t + 1] - U[-2, t + 1])
    C[-1, t + 1] = C_new_boundary - C_boundary_adjustment_term




# Вычисление деформации как производной смещения
E = np.zeros((M + 1, T + 1))  # Массив для деформаций

for m in range(1, M):  # Для всех узлов, кроме краевых
    for t in range(T + 1):  # Для всех моментов времени
        E[m, t] = (U[m + 1, t] - U[m - 1, t]) / (2 * Δh)

# Обработка краевых узлов
E[0, :] = (U[1, :] - U[0, :]) / Δh
E[-1, :] = (U[-1, :] - U[-2, :]) / Δh

# Вычисление напряжений
Sigma = np.zeros((M + 1, T + 1))  # Массив для напряжений
```

```
for t in range(T):
    for m in range(1, M):
        div_U = (U[m + 1, t + 1] - U[m - 1, t + 1]) / (2 * Δh)
        epsilon = div_U  # В одномерном случае divergence U и ε совпадают
        Sigma[m, t + 1] = λ * div_U + 2 * μ * epsilon


for t in range(T):
    for m in range(1, M):
        # Додамо обчислення напруження
        div_U = (U[m+1, t+1] - U[m-1, t+1]) / (2*Δh)
        epsilon = div_U  # У одномірному випадку divergence U і ε збігаються
        Sigma[m, t+1] = λ * div_U + 2 * μ * epsilon
# Вычисление деформации как производной смещения
E = np.zeros((M+1, T+1))  # массив для хранения деформаций
for m in range(1, M):  # для всех узлов, кроме краевых
    E[m, :] = (U[m+1, :] - U[m-1, :]) / (2*Δh)

# Краевые узлы обрабатываются отдельно
E[0, :] = (U[1, :] - U[0, :]) / Δh
E[-1, :] = (U[-1, :] - U[-2, :]) / Δh

# График деформации во времени для всех узлов

# Создаем списки для хранения датафреймов
dfs_C = []
dfs_U = []
dfs_E = []
dfs_Sigma = []

# Создаем датафреймы и индексы для каждого M
for i in range(M+1):
    index_values = ['[{},{}]'.format(i, j) for j in range(T+1)]

    df_C_i = pd.DataFrame(C[i,:], columns=['Value'])
    df_U_i = pd.DataFrame(U[i,:], columns=['Value'])
```

```
        df_E_i = pd.DataFrame(E[i,:], columns=['Value'])
        df_Sigma_i = pd.DataFrame(Sigma[i,:], columns=['Value'])

        df_C_i.index = ['C' + idx for idx in index_values]
        df_U_i.index = ['U' + idx for idx in index_values]
        df_E_i.index = ['E' + idx for idx in index_values]
        df_Sigma_i.index = ['Sigma' + idx for idx in index_values]

        dfs_C.append(df_C_i)
        dfs_U.append(df_U_i)
        dfs_E.append(df_E_i)
        dfs_Sigma.append(df_Sigma_i)

    # Объединяем все датафреймы в один
    df_C = pd.concat(dfs_C)
    df_U = pd.concat(dfs_U)
    df_E = pd.concat(dfs_E)
    df_Sigma = pd.concat(dfs_Sigma)

    # Теперь можно сохранить датафреймы в Excel
    with pd.ExcelWriter('output_{}.xlsx'.format(datetime.datetime.now().strftime("%Y-%m-%d %H-%M-%S")),
engine='xlsxwriter') as writer:
        df_C.to_excel(writer, sheet_name='Sheet_C')
        df_U.to_excel(writer, sheet_name='Sheet_U')
        df_E.to_excel(writer, sheet_name='Sheet_E')
        df_Sigma.to_excel(writer, sheet_name='Sheet_Sigma')

    fig, axs = plt.subplots(4, 2, figsize=(20, 40))  # создаем 8 подграфиков
    t = np.linspace(0, T*Δt, T+1)
    # Графік концентрації водню в різні моменти часу
    x = np.linspace(0, (Δh*M), M+1)
    if T < 5:
```

```
        step = 1
    else:
        step = int((T+1)/5)
    for t in range(0, T+1, step):
        # ваш код здесь


        if len(x) == C[:, t].shape[0]:
            axs[0,0].plot(x, C[:, t], label=f'Крок часу {t}')
        else:
            print(f"Неможливо побудувати графік для кроку часу {t} через невідповідність розмірів: len(x) =
{len(x)}, C[:, t].shape[0] = {C[:, t].shape[0 ]}")


    axs[0,0].set_title('Концентрація водню з часом')
    axs[0,0].set_xlabel('Відстань по пластині')
    axs[0,0].set_ylabel('Концентрація')


    axs[0,0].grid(True)


    # Графік концентрації водню з часом для всіх вузлів
    t = np.linspace(0, T*Δt, T+1)  # масив часу
    for m in range(M+1):  # цикл по всіх вузлах
        if len(t) == C[m, :].shape[0]:
            axs[0,1].plot(t, C[m, :], label=f'Вузол {m}')
        else:
            print(f"Неможливо побудувати графік для вузла {m} через невідповідність розмірів: len(t) = {len(t)},
C[m, :].shape[0] = {C[m, :].shape[0] }")


    axs[0,1].set_title('Концентрація водню з часом для всіх вузлів')
    axs[0,1].set_xlabel('Час')
    axs[0,1].set_ylabel('Концентрація')


    axs[0,1].grid(True)
    t = np.linspace(0, T*Δt, T+1)
    # Графік напруження з часом
    x = np.linspace(0, (Δh*M), M+1)
    step = int((T+1)/5) if T >= 5 else 1
```

```
    for t in range(0, T+1, step):  # кожні 20% часу
    # ...

        if len(x) == Sigma[:, t].shape[0]:
            axs[1,0].plot(x, Sigma[:, t], label=f'Крок часу {t}')
        else:
            print(f"Неможливо побудувати графік для кроку часу {t} через невідповідність розмірів: len(x) = {len(x)}, Sigma[:, t].shape[0] = {Sigma[:, t].shape[0 ]}")
    axs[1,0].set_title('Напруження з часом')
    axs[1,0].set_xlabel('Відстань по пластині')
    axs[1,0].set_ylabel('Напруга')

    axs[1,0].grid(True)
    t = np.linspace(0, T*Δt, T+1)
    # Напруження з часом для всіх вузлів
    x = np.linspace(0, (Δh*M), M+1)
    if T < 5:
        step = 1
    else:
        step = int((T+1)/5)
    for t in range(0, T+1, step):
        # ваш код здесь
        if len(x) == U[:, t].shape[0]:
            axs[2,0].plot(x, U[:, t], label=f'Часовий крок {t}')
        else:
            print(f"Не вдається побудувати графік для кроку часу {t} через невідповідність розмірів: len(x) = {len(x)}, U[:, t].shape[0] = {U[:, t].shape[0 ]}")


    axs[2,0].set_title('Зміщення з часом')
    axs[2,0].set_xlabel('Відстань по пластині')
    axs[2,0].set_ylabel('Зміщення')
    axs[2,0].legend(loc='center left', bbox_to_anchor=(1, 0.5))  # додаємо легенду
    axs[2,0].grid(True)  # додаємо сітку

    # Графік напруження з часом для всіх вузлів
```

```
    t = np.linspace(0, T*Δt, T+1)  # масив часу
    for m in range(M+1):  # цикл по всіх вузлах
        if len(t) == Sigma[m, :].shape[0]:
            axs[1,1].plot(t, Sigma[m, :], label=f'Вузол {m}')
        else:
            print(f"Неможливо побудувати графік для вузла {m} через невідповідність розмірів: len(t) = {len(t)}, Sigma[m, :].shape[0] = {Sigma[m, :].shape[0] }")

    axs[1,1].set_title('Часове напруження для всіх вузлів')
    axs[1,1].set_xlabel('Час')
    axs[1,1].set_ylabel('Напруга')

    axs[1,1].grid(True)

    # Графік зміщення з часом для всіх вузлів
    t = np.linspace(0, T*Δt, T+1)  # масив часу
    for m in range(M+1):  # цикл по всіх вузлах
        if len(t) == U[m, :].shape[0]:
            axs[2,1].plot(t, U[m, :], label=f'Вузол {m}')
        else:
            print(f"Неможливо побудувати графік для вузла {m} через невідповідність розмірів: len(t) = {len(t)}, U[m, :].shape[0] = {U[m, :].shape[0] }")

    axs[2,1].set_title('Зміщення з часом для всіх вузлів')
    axs[2,1].set_xlabel('Час')
    axs[2,1].set_ylabel('Зсув')
    axs[2,1].legend(loc='center left', bbox_to_anchor=(1, 0.5))
    axs[2,1].grid(True)
    t = np.linspace(0, T*Δt, T+1)
    # Графік деформації в різні моменти часу
    x = np.linspace(0, (Δh*M), M+1)
    if T < 5:
        step = 1
    else:
        step = int((T+1)/5)
    for t in range(0, T+1, step):
```

```
    # ваш код здесь
        if len(x) == E[:, t].shape[0]:
            axs[3,0].plot(x, E[:, t], label=f'Крок часу {t}')
        else:
            print(f"Неможливо побудувати графік для кроку часу {t} через невідповідність розмірів: len(x) =
{len(x)}, E[:, t].shape[0] = {E[:, t].shape[0 ]}")


    axs[3,0].set_title('Деформація на всій пластині з часом')
    axs[3,0].set_xlabel('Відстань по пластині')
    axs[3,0].set_ylabel('Деформація')


    axs[3,0].grid(True)


    # Графік деформації з часом для всіх вузлів
    t = np.linspace(0, T*Δt, T+1)  # масив часу
    for m in range(M+1):  # цикл по всіх вузлах
        if len(t) == E[m, :].shape[0]:
            axs[3,1].plot(t, E[m, :], label=f'Вузол {m}')
        else:
            print(f"Не вдається побудувати графік для вузла {m} через невідповідність розмірів: len(t) = {len(t)},
E[m, :].shape[0] = {E[m, :].shape[0] }")
    axs[3,1].set_title('Деформація з часом для всіх вузлів')
    axs[3,1].set_xlabel('Час')
    axs[3,1].set_ylabel('Деформація')


    axs[3,1].grid(True)


    plt.subplots_adjust(left=0.2, right=0.8, bottom=0.1, top=0.9, wspace=0.8, hspace=0.6)
    fig.savefig(pdf_pages, format='pdf')
    for widget in frame.winfo_children():
        widget.destroy()


    # Отображение графика в tkinter
    canvas = FigureCanvasTkAgg(fig, master=frame)
    canvas.draw()
    canvas.get_tk_widget().pack()
```

```
    plt.close(fig)
    pdf_pages.close()
def main_menu():
    root = tk.Tk()
    root.title("Меню")



    frame = tk.Frame(root)
    frame.pack(side=tk.LEFT)  # окно ввода данных слева

    plot_frame = tk.Frame(root)
    plot_frame.pack(side=tk.RIGHT)  # окно графиков справа

    # Текстові поля для вводу початкових даних
    D_label = tk.Label(frame, text="D:")
    D_label.grid(row=0, column=0)
    D_entry = tk.Entry(frame)
    D_entry.grid(row=0, column=1)

    v_label = tk.Label(frame, text="v:")
    v_label.grid(row=1, column=0)
    v_entry = tk.Entry(frame)
    v_entry.grid(row=1, column=1)

    E_label = tk.Label(frame, text="E:")
    E_label.grid(row=2, column=0)
    E_entry = tk.Entry(frame)
    E_entry.grid(row=2, column=1)




    Bc_label = tk.Label(frame, text="Bc:")
    Bc_label.grid(row=4, column=0)
    Bc_entry = tk.Entry(frame)
    Bc_entry.grid(row=4, column=1)
```

```
    c0_label = tk.Label(frame, text="c0:")
    c0_label.grid(row=5, column=0)
    c0_entry = tk.Entry(frame)
    c0_entry.grid(row=5, column=1)

    a_label = tk.Label(frame, text="a:")
    a_label.grid(row=6, column=0)
    a_entry = tk.Entry(frame)
    a_entry.grid(row=6, column=1)

    M_label = tk.Label(frame, text="M:")
    M_label.grid(row=7, column=0)
    M_entry = tk.Entry(frame)
    M_entry.grid(row=7, column=1)

    T_label = tk.Label(frame, text="T:")
    T_label.grid(row=8, column=0)
    T_entry = tk.Entry(frame)
    T_entry.grid(row=8, column=1)

    Δh_label = tk.Label(frame, text="Δh:")
    Δh_label.grid(row=9, column=0)
    Δh_entry = tk.Entry(frame)
    Δh_entry.grid(row=9, column=1)

    Δt_label = tk.Label(frame, text="Δt:")
    Δt_label.grid(row=10, column=0)
    Δt_entry = tk.Entry(frame)
    Δt_entry.grid(row=10, column=1)



    def submit():
        nonlocal D_entry, v_entry, E_entry,  Bc_entry, c0_entry, a_entry, M_entry, T_entry, Δh_entry, Δt_entry
        D, v, E,  Bc, c0, a, M, T, Δh, Δt = read_input(D_entry, v_entry, E_entry,  Bc_entry, c0_entry, a_entry,
M_entry, T_entry, Δh_entry, Δt_entry)
```

```
        plot_all(plot_frame, D, v, E,  Bc, c0, a, M, T, Δh, Δt)

    button_submit = tk.Button(frame, text="Оновити", command=submit)
    button_submit.grid(row=11, column=0, columnspan=2)

    button_quit = tk.Button(frame, text="Вихід", command=root.quit)
    button_quit.grid(row=12, column=0, columnspan=2)

    root.mainloop()

if __name__ == "__main__":
    main_menu()
```

Додаток В

Презентація

МОДЕЛЮВАННЯ ДАТЧИКА ВИЯВЛЕННЯ ВИТОКУ ВОДНЮ.

Спецчастина: «Розробка програмних модулів для моделювання роботи датчиків та дослідницького інтерфейсу»

Спеціальність: "122 Комп'ютерні науки"

Автор: Кузьменко Микита Андрійович.
Керівник: Олена ЛЮБИМЕНКО

Актуальність

- Розробка програмних модулів для моделювання роботи датчика та інтерфейсу відповідає сучасним технологічним тенденціям.
- Розробка нових модулів адаптує датчики до різних платформ, сприяючи їх широкому використанню в різних галузях промисловості.
- Дана тема розвиває наукову базу знань і сприяє підготовці фахівців, що відповідають сучасним вимогам ринку праці.

Цілі та завдання проєкту

- **Об'єктом дослідження** є процес моделювання роботи сенсору контролю витоку водню в реальних умовах.
- **Предметом дослідження** є математична модель для контролю витоку водню.
- **Метою** цієї **роботи** є розробка програмних модулів для моделювання роботи сенсору та створення інтерфейсу з виводом результатів дослідження.
- **Завдання** полягає в розробці математичної моделі для моделювання роботи сенсору витоку водню, використовуючи мову програмування Python та такі бібліотеки Python: numpy, matplotlib, та tkinter.
- **Практичне значення** полягає в створенні графічного інтерфейсу користувача для візуалізації результатів моделювання та аналізу даних.

Аналіз сфери використання сенсорів водню

•**Область досліджень**: Вивчення впливу водню на паладій - активна та динамічна область, яка постійно розвивається і вдосконалюється.

•**Основа для дослідження**: Існуючі математичні моделі, що описують дифузію водню в металах, слугують цінною основою для нашого дослідження.

•**Адаптація методів розв'язання**: Нам потрібно адаптувати існуючі методи розв'язання, такі як метод скінченних різниць, до наших конкретних умов та вимог.

•**Оптимізація програмних інструментів**: Необхідно адаптувати програмні інструменти, використовувані для моделювання водневої дифузії, до наших вимог і обмежень.

•**Подальше дослідження**: Для забезпечення валідності та релевантності наших результатів, ми плануємо провести детальне дослідження відповідно до специфіки нашого проекту.

Проєктування програмних модулів

•**Ціль розробки**: Моделювання процесу дифузії водню в пластині паладію, що включає в себе зміни концентрації водню з часом і залежність від еластичних напружень у металі.

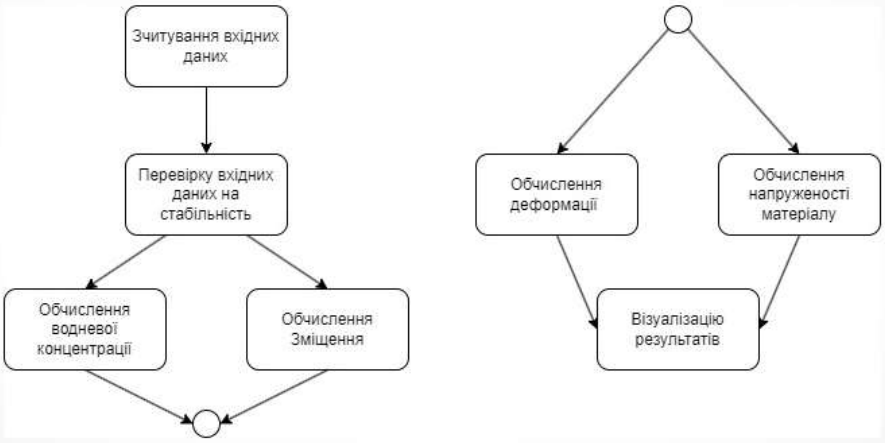
•**Основа моделі**: Модель базується на одновимірних рівняннях водневої еластичності, які враховують просторово-часові зміни концентрації водню та еластичних напружень.

•**Розрахунковий метод**: Використовується метод скінченних різниць, що передбачає розбиття пластини на велику кількість малих елементів і розрахунок змін в концентрації водню в кожному з них.

•**Крайові умови**: Застосовано різні крайові умови для відтворення різних сценаріїв, що включають постійне насичення воднем, відсутність водню, та насичення воднем, що змінюється в часі.

•**Розв'язання системи рівнянь**: Використовується метод Гауса для розв'язання системи лінійних рівнянь. Однак, з урахуванням об'єму рівнянь, це може вимагати значних обчислювальних ресурсів.

План програмного продукту



На шляху до створення програми ряд подальших досліджень

- Удосконалення числового методу
- Оптимізація виконання коду для прискорення обчислень
- Розширення функціональності для обробки більш складних моделей
- Додавання додаткових опцій для користувача

Математична модель

Диференціальне рівняння

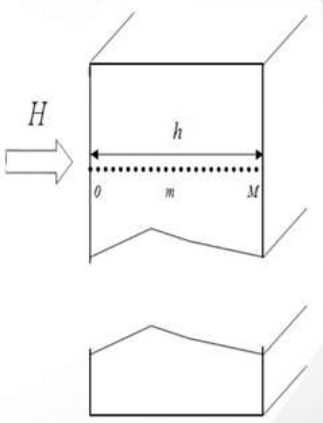
$$\begin{cases} \dfrac{\partial^2 c}{\partial x^2} + G\dfrac{\partial c}{\partial t} Hc \dfrac{\partial}{\partial x}\dfrac{\partial u_x}{\partial t} = 0 \\ S\dfrac{\partial^2 u_x}{\partial x^2} + Q\dfrac{\partial c}{\partial x} = 0 \end{cases}$$

Де $G = \frac{1}{D}$,

$Q = G(3\lambda + 2\mu)w$,

$Y = QB_c$,

$S = \lambda + 2\mu$,



Початкові та граничні умови для меж пластини

$$c(0,t) = c_0(1 - e^{-at})$$

$$\frac{dc}{dx}_{|x=h} = 0$$

Математична модель

Внутрішні напруження

$$\frac{1}{u}\int_0^h \left(-\frac{\partial u_x}{\partial x}\right)dx - \frac{w}{1-2u}\int_0^h (c(x)-c_h)dx - 0,$$

$$\frac{1}{u}\int_0^h \left(-\frac{\partial u_x}{\partial x}\right)xdx - \frac{w}{1-2u}\int_0^h (c(x)-c_h)xdx - 0,$$

Початкові умови

$c(x,0) = 0, \qquad u(x,0) = 0$ для $0 \le x \le 0$

Математична модель записана за допомогою кінцевих різниць

Перетворене диференціальні рівняння

- $$\begin{cases} \left(\frac{1}{\Delta h}\right)^2 \left(c_{m+1}^2 - 2c_m^p + c_{m-1}^p\right) + \left(\frac{G}{\Delta t}\right)\left(c_m^{p+1} - c_m^p\right) + \left(\frac{H}{2\Delta h \Delta t}\right)c_m^p \\ \left(u_{m+1}^{p+1} - u_{m-1}^{p+1} + u_{m-1}^p\right) = 0 \\ \\ \frac{S}{(\Delta h)^2}\left(u_{m+1}^{p+1} - 2u_m^{p+1} + u_{m-1}^{p+1}\right) + \left(\frac{Q}{2\Delta h}\right)\left(c_{m+1}^{p+1} - c_{m-1}^{p+1}\right) = 0 \end{cases}$$

Для меж пластини

- $c_0^{p+1} = c_0^p = c_0$
- $c_1^{p+1} = c_0(1 - e^{-a(p+1)\Delta t})$
- $c_M^{p+1} - c_{M-1}^{p+1} = 0$

Математична модель записана за допомогою кінцевих різниць

Перетворене рівняння внутрішньої напруження

$$\frac{1}{u}\left(u_1^{p+1} - u_M^{p+1}\right) - \frac{w\Delta h}{1-2u}\sum_{m=1}^{M} c_m^{p+1} + \frac{wh}{1-2u}c_M^{p+1} = 0,$$

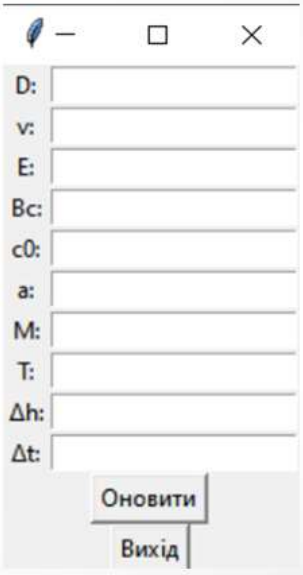
$$-\frac{\Delta h}{u}\sum_{m=1}^{M-1}\left(u_{m+1}^{p+1} - u_m^{p+1}\right)m - \frac{w(\Delta h)^2}{1-2u}\sum_{m=1}^{M} c_m^{p+1}m + \frac{wh^2}{2(1-2u)}c_M^{p+1} = 0,$$

Початкові умови

$$c_m^0 = 0, \qquad u_m^0 = 0, \text{где } m = \overline{1, M}$$

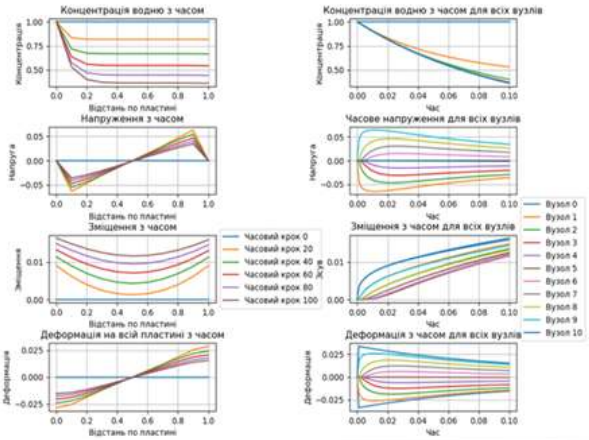
Інтерфейс користувача

Користувальницький інтерфейс, побудований з використанням бібліотеки tkinter, забезпечує простий та інтуїтивно зрозумілий спосіб введення даних користувача для імітації роботи датчика виявлення витоку водню.

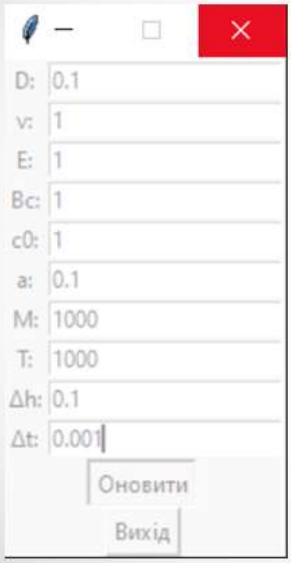
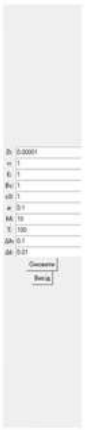


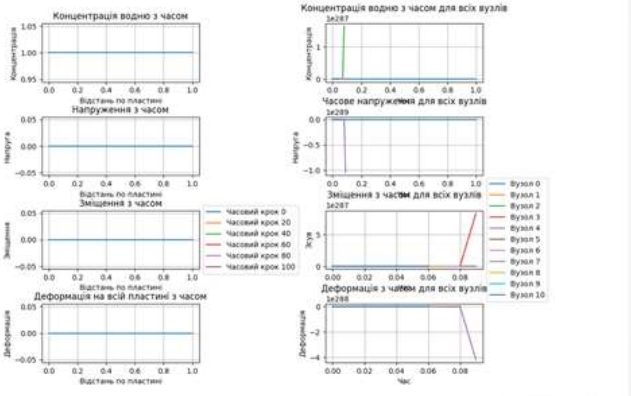
Візуалізація

- концентрація водню уздовж пластини в різний час;
- концентрація водню у всіх вузлах за часом
- розподіл напружень уздовж пластини в різ моменти часу;
- напруги на всіх вузлах в часі;
- розподіл переміщень уздовж плити в різні моменти часу;
- зміщення у всіх вузлах за часом;
- розподіл деформацій уздовж пластини в різні моменти часу;
- деформація у всіх вузлах за часом.



Тестування





Висновки по роботі

- Розробка математичної моделі:
- Розробка інтерфейсу користувача
- Візуалізація результатів
- Реалізація програмних модулів
- Подальші дослідження

- ❑ Успішно розроблена математична модель для імітації роботи датчика виявлення витоку водню.
- ❑ Створено інтуїтивно зрозумілий та простий у використанні інтерфейс для введення користувачем.
- ❑ Результати моделювання були успішно візуалізовані та представлені у зрозумілому та доступному для аналізу форматі.
- ❑ Використовуючи мову програмування Python та використовуючи бібліотеки numpy, matplotlib та tkinter, були реалізовані програмні модулі для моделювання роботи значення ога та демонстрації результатів.