МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Сумський державний університет
Факультет технічних систем та енергоефективних технологій
Кафедра комп'ютерної механіки імені Володимира Марцинковського

«До захисту допущено»
Завідувач кафедри
_____ Андрій ЗАГОРУЛЬКО
(підпис)
____ _____ 2023 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня бакалавр

зі спеціальності 131 Прикладна механіка, освітньо-професійної програми «Комп'ютерний інжиніринг в механіці», на тему: Розроблення комп'ютерної програми розрахунку власних частот ротора на основі мови програмування Python.

Здобувача групи КМ-91-1 БУРИМА Миколи Олексійовича.

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

_____ Микола БУРИМ
(підпис)

Керівник: професор, д.т.н., професор Іван ПАВЛЕНКО _____
(підпис)

Суми – 2023

АНОТАЦІЯ

Структура та обсяг кваліфікаційної роботи бакалавра: складається зі вступу, 4 розділів, загальних висновків, списку використаних джерел, що містить 17 найменувань. Загальний обсяг бакалаврської роботи становить 58 стор., у тому числі 2 таблиці, 3 рисунки, списку використаних джерел обсягом 2 сторінки.

Ротор, вібродіагностування, модальний аналіз, технічний стан, вібраційний процес та жорсткість.

Об’єктом дослідження є коливальні процеси, що відбуваються в роторах та вплив ефекту резонансу.

Метою роботи є розроблення комп’ютерної програми для оцінювання значень власних частот ротора.

У першому розділі коротко представлений огляд літературних джерел про модальний аналіз ротора та основні можливості мови програмування Python. У другому розділі наведена інформація про використані бібліотеки та вимоги до програми. У третьому розділі описано процес розроблення та основні функції. У останньому розділі даної роботи представлено результати розрахунку та перевірка його надійності. У кінці роботи сформульовані висновки за темою дослідження.

Робота виконувалась у рамках НДР № БФ/26-2021 на Виконання завдань перспективного плану розвитку наукового напряму «Технічні науки» Сумського державного університету (держреєстрація № 0121U112684).

ЗМІСТ

ВСТУП

Однією із найважливіших проблем, яку потрібно вирішити при розробці ротора для роторної машини, є розрахунок його власних частот. Власні частоти роторів залежать від механічних характеристик матеріалу, або матеріалів, та геометричних розмірів. Такі розрахунки є комплексними, що добре підлягає програмуванню.

Критичність розрахунку випливає із ефекту резонансу, який виникає при наближенні значення збуджуючих коливань із зовні до власної частоти системи. Іншими словами: «Власні частоти ротора - це частоти, при яких ротор може приймати резонансні коливання»

Ефект резонансу може мати наступні наслідки для ротора:

- Збільшення амплітуди коливань: Під впливом резонансу амплітуда коливань ротора може значно зрости. Це може спричинити до механічної напруги, зносу та пошкодження компонентів ротора.
- Зміна динамічних характеристик: Резонанс може призвести до зміни динамічних характеристик ротора, таких як частоти власних коливань та форми мод. Це може вплинути на стійкість ротора та його роботу.
- Пошкодження системи: Якщо амплітуда коливань стає надто великою, це може призвести до механічного пошкодження системи, зламу компонентів або втрати ефективності роботи ротора.

Метою цієї роботи є розробка програми для чисельного розрахунку власних частот ротора, використовуючи файл вхідних даних та подальшим виводом результатів у вигляді тексту, графіку та графічним відображенням ротора із вхідних даних. Мовою програмування було обрано Python.

Python - це високорівнева мова програмування, яка працює у режимі інтерпретації. Вона заснована на об'єктно-орієнтованому підході і має гнучку семантику, що дозволяє змінювати програму під час її виконання. [1]

Python пропонує високорівневі структури даних, динамічну типізацію та динамічне зв'язування. Ці особливості роблять його ідеальним для швидкої розробки додатків, написання сценаріїв та поєднання існуючих компонентів. Простий та легкий для вивчення синтаксис Python забезпечує легку розуміння коду і зменшує вартість обслуговування програм. Python також підтримує модулі та пакети, що сприяє модульності програми і повторному використанню коду. Інтерпретатор Python та його велика стандартна бібліотека доступні безкоштовно для всіх основних платформ і можуть вільно використовуватися.

Переваги Python, як мови програмування для наукових обчислень [2]:

- Широкі наукові обчислювальні бібліотеки
- Добре продумана мова, що дозволяє писати код, який легко читати
- Наявність великої кількості бібліотек, що не стосуються наукових обчислень (веб-сервер, доступ до послідовного порту тощо)
- Безкоштовне програмне забезпечення з відкритим вихідним кодом, широко розповсюджене та живе співтовариство.
- Різноманітність потужних середовищ для роботи, таких як «IPython», «Spyder», «Jupyter notebooks», «Pycharm», «Visual Studio Code»

1 АНАЛІЗ ЛІТЕРАТУРИ

1.1 Результати дослідження математичної моделі коливань ротора

Згідно до статей [3] та [4] модель ротора описується диференційним рівнянням:

$$M\ddot{x} + D\dot{x} + Kx = P(t) \tag{1.1}$$

Де,

M – узагальнена матриця імпульсу;

D – узагальнена матриця демпфування;

K – узагальнена матриця жорсткості;

$\ddot{x}, \dot{x}, x$ – узагальнений вектор зміщення, швидкості та прискорення;

P – узагальнений вектор зовнішнього збудження;

t – час;

Для аналітичного розрахунку достатньо розробити строщену модель ротора, власна частота якої буде розраховуватися за формулами, як було виведено у статті [3]:

- Для ротора із вільними кінцями:

$$f = \frac{\pi}{2} \cdot n^2 \cdot \sqrt{\frac{g \cdot E \cdot I}{w \cdot L^4}} \text{ [Гц]} \tag{1.2}$$

- Для ротора із жорстко закріпленими кінцями:

$$f = \frac{\pi}{2} \cdot \left(n + \frac{1}{2}\right)^2 \cdot \sqrt{\frac{g \cdot E \cdot I}{w \cdot L^4}} \text{ [Гц]} \tag{1.3}$$

- Для консольно закріпленого ротора, із підшипником на одному кінці

$$f = \frac{\pi}{2} \cdot \left(n - \frac{1}{2}\right)^2 \cdot \sqrt{\frac{g \cdot E \cdot I}{w \cdot L^4}} \text{ [Гц]} \quad (1.4)$$

Де,

n – номер мод (власної частоти)

g – прискорення вільного падіння, $\frac{\text{м}}{\text{с}^2}$

E – модуль Юнга для матеріалу секції ротора, ГПа

I – момент інерції, м^4

w – жорсткість, $\frac{\text{Н}}{\text{м}}$

L – довжина валу, м

При збуджені валу частотою близькою до його власної частоти, виникає ефект резонансу, що спричиняє деформацію валу. У залежності від номеру власної частоти, вал починає приймає певну форму, що називається мод. [5][6]

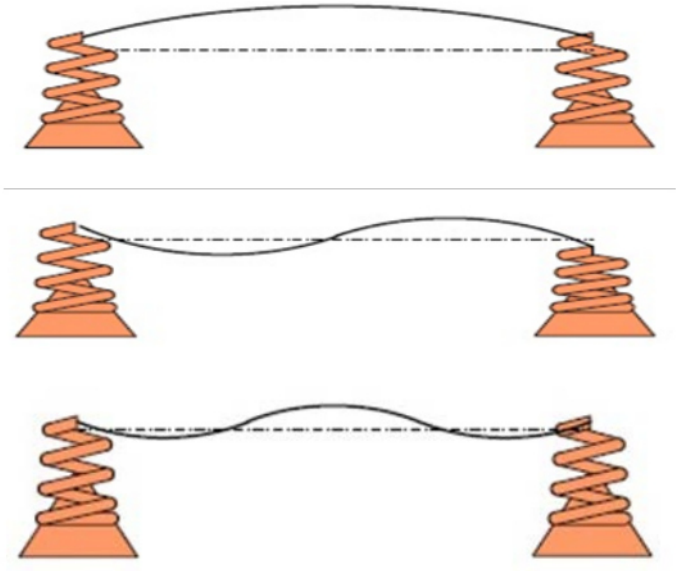

Рисунок 1.1 – Форми трьох мод для еластичного валу

1.2 Методи дослідження вібраційного стану ротора

Дослідження вібраційного стану ротора включає в себе використання різних методів для вимірювання, аналізу та оцінки вібрацій, що виникають під час роботи ротора. Одним із них є модальний аналіз, що описується вище. Результати модального аналізу отримуються за допомогою аналітичних, або чисельних розрахунків [3][7].

Також, для проведення вібраційного аналізу використовують метод лазерної віброметрії, про дослідження якої йдеться у статті [8]. Автори вказують на недостатню кількість досліджень у цій сфері та на обмеження методу. Наприклад, у статті наводиться інформація про ступінь крос-чутливості, який впливає на результати дослідження. Крос-чутливість це поняттям вказує наскільки вимірювальний прилад чутливий до рухів або впливів, відмінних від основного руху, який потрібно виміряти. крос-чутливість відображає, як датчик або пристрій вимірювання реагує на рухи або впливи, які відбуваються в інших напрямках або площинах, крім основного радіального руху. Це можуть бути напрямки, такі як осьові коливання, зміна швидкості або гнучкі коливання.

1.3 Аналіз існуючих програмних засобів

Для ефективного оцінювання вібраційного стану ротора і виявлення потенційних проблем широко використовують різноманітні програмні засоби. Ці засоби можуть бути як платні (ANSYS [9], MATLAB [11]) так і безкоштовні (Python [1], Octave), що допомагають вимірювати, аналізувати та візуалізувати дані вібрацій, що дозволяє здійснювати раціональну діагностику та планувати попереджувальний технічний обслуговування.

Проте при використанні ANSYS для аналізу вібраційного стану ротора можуть виникати такі проблеми:

1. Складність моделювання.
2. Великий обсяг обчислень.
3. Ліцензійні витрати

Python має переваги перед ANSYS у таких аспектах:

- Відкритий код: Python має відкритий код, що дозволяє змінювати та настроювати його за потребами.
- Багатофункціональність: Python має широкий вибір бібліотек, що полегшують аналіз даних та визначення вібраційного стану.
- Оптимізація: Python не потребує багато обчислювальних потужностей, якщо код написано грамотно.
- Спільнота та ресурси: Python має велику спільноту користувачів та доступ до багатої документації та ресурсів.

Використання Python для оцінювання вібраційного стану ротора дозволяє отримати більш гнучкі та настроювані рішення.

1.4 Висновки до першого розділу

У результаті аналізу літератури можна зробити висновки, що розробка полегшеного програмного забезпечення для розрахунку модального аналізу ротора є важливою проблемою, яка потребує вирішення. Наведена математична модель дозволяє запрограмувати її обчислення без використання методу скінченних елементів та програмування операцій вищої математики.

Для отримання більш точних результатів, можна скористатися програмним комплексом ANSYS. Використання ANSYS не завжди є доречним у зв’язку із складністю моделювання, обмеженими обчислювальними можливостями, вартістю та ліцензійним обмеженням, та необхідністю володіння навичками роботи.

2 МЕТОДОЛОГІЯ ДОСЛІДЖЕННЯ

2.1 Визначення вимог до програми

1. Функціональні вимоги:
    - Зчитування вхідних даних з файлу у форматі *.xlsx.
    - Перевірка наявності файлу перед зчитуванням.
    - Обробка та підготовка даних для подальшого аналізу.
    - Розрахунок власних частот ротора на основі заданих параметрів.
    - Виведення результатів у вигляді графіку для візуалізації власних частот.
2. Нефункціональні вимоги:
    - Програма повинна бути ефективною з точки зору обробки великих обсягів даних.
    - Надійність програми: вона повинна бути стійкою до помилок та відповідати вимогам точності обчислень.
    - Код програми повинен бути добре організованим

2.2 Опис обраних бібліотек

У даному розділі роботи буде проведена аргументація та обґрунтування вибору конкретних бібліотек для розроблення комп'ютерної програми розрахунку власних частот ротора та виведення результатів у вигляді графіку. Основними вимогами до програмного продукту є точність розрахунків, швидкість обчислень, зручний вивід результатів та графічне представлення даних. Для досягнення цих

цілей було вивчено різні бібліотеки та інструменти, і на основі аналізу їх можливостей та переваг були обрані наступні бібліотеки.

Вхідні данні будуть отримуватися із файлу із розширенням ‘.xlsx’, отже потрібна бібліотека «pathlib» а також «os» із модулем «os.path» для перевірки наявності файлу. Це дозволить уникнути можливих помилок, пов'язаних з відсутністю файлу або неправильним шляхом до нього.

На етапі розрахунків будуть використовуватися бібліотеки для роботи із масивами даних та подальшої їх обробки, а саме:

- NumPy - це потужна бібліотека для наукових обчислень у мові програмування Python. [11] Вона надає високопродуктивні масиви та функції для роботи з ними. NumPy дозволяє ефективно виконувати математичні операції над масивами даних, що є основою для обчислення власних частот ротора.
- SciPy - це бібліотека, яка розширює функціональність NumPy, надаючи додаткові можливості для наукових обчислень. [12] При проведенні розрахунків буде використано тільки модуль «scipy.interpolate». Модуль interpolate надає різні методи інтерполяції, такі як лінійна, квадратична або кубічна інтерполяція.

Для візуалізації результатів розрахунку буде використовуватися бібліотека «Matplotlib».

Matplotlib - це бібліотека для візуалізації даних у мові програмування Python. Вона надає різноманітні інструменти для створення графіків, діаграм, гістограм та інших видів візуалізації. [13] Завдяки Matplotlib можна легко вивести результати розрахунків у вигляді графіків, що дозволить зрозуміло представити власті частоти ротора та їх залежність від різних параметрів.

3 ПОБУДОВА ПРОГРАМИ

3.1 Основні функції програми

Функція - це блок коду, який виконує певну задачу і може бути викликаний з інших частин програми. Вона дозволяє організувати код у логічні блоки, які можна використовувати повторно.

У Python функцію можна створити за допомогою ключового слова «def», за яким слідує ім'я функції, дужки з аргументами і двокрапка. Тіло функції потрібно відступити від ключового слова def на один рівень.

```
def getfile():
    while True:
        file = input('\nEnter file name *.xlsx: ')
        filename = file + '.xlsx'
        path = pathlib.Path(filename)
        if path.exists():
            return path
        else:
            print('Incorrect file name.')

file = getfile()
```

Визначення функції під назвою «getfile» що починається безкінечним циклом «while True», та слугує для отримання назви файлу від користувача із розширенням «.xlsx» з подальшим його пошуком. Якщо файл не знайдено в системі, тоді користувач отримує повідомлення «Incorrect file name», а цикл повертається на початок. Якщо файл існує, функція повертає об’єкт «path» до головної програми.

```
def getnumber():
    while True:
        number = input("\nEnter a number of frequencies: ")
        if number.isdigit():
            return number
        else:
            print('Incorrect number.')

num = int(getnumber())
```

Визначення функції під назвою «getnumber» що починається безкінечним циклом «while True», та слугує для отримання кількості частот для розрахунку від користувача. У випадку якщо введені дані не є цифрою, користувач отримує повідомлення «Incorrect number». В іншому випадку функція повертає об'єкт «number» до головної програми.

3.2 Розрахунок власних частот

```
A = pd.read_excel(file)
A = np.asarray(A)
A = A[:,1:]
```

Перший рядок коду за допомогою функції read_excel виконує перетворення Excel файлу в об'єкт DataFrame бібліотеки Pandas. [15]

Для більш широких можливостей для обробки та аналізу даних наступним кроком DataFrame у масив NumPy із подальшим відкиданням першого стовбця, тобто стовбця із індексом 0.

```
n, cols = A.shape
C = np.zeros((2*(n+1),2*(n+1)))
M = np.zeros((2*(n+1),2*(n+1)))
L, D, d, m0, I_m, k, alpha, dens, E = A[:,0], A[:,1], A[:,2], A[:,3], A[:,4],
A[:,5], A[:,6], A[:,7], A[:,8]

I, m = [], []
for i in range(n):
    I.append(pi*(D[i]**4-d[i]**4)/64)
    m.append(dens[i]*L[i]*pi*(D[i]**2-d[i]**2)/4)
```

Метод «shape» використовується для отримання розмірності матриці у бібліотеці NumPy. Одержані значення кількості рядків та стовбців записуються до змінних «n» та «cols» відповідно.

Із метою проведення обчислень створюється дві квадратні нульові матриці «C» та «M» розмірністю $(2 \cdot (n + 1))$

За допомогою індексування масиву «A», значення з різних стовпців «A» (від 0 до 8) копіюються в окремі змінні «L» (довжина), «D» (зовнішній діаметр), «d» (внутрішній діаметр), «m0», «I_m» (момент інерції), «k» (жорсткість), «alpha», «dens» (густина) та «E» (модуль Юнга).

```
    for i in range(n):

    C[2*i,2*i] += 12*E[i]*I[i]/L[i]**3 + k[i]
    C[2*i,2*i+1] += 6*E[i]*I[i]/L[i]**2
    C[2*i,2*i+2] += -12*E[i]*I[i]/L[i]**3
    C[2*i,2*i+3] += 6*E[i]*I[i]/L[i]**2

    C[2*i+1,2*i] += 6*E[i]*I[i]/L[i]**2
    C[2*i+1,2*i+1] += 4*E[i]*I[i]/L[i]
    C[2*i+1,2*i+2] += -6*E[i]*I[i]/L[i]**2
    C[2*i+1,2*i+3] += 2*E[i]*I[i]/L[i]

    C[2*i+2,2*i] += -12*E[i]*I[i]/L[i]**3
    C[2*i+2,2*i+1] += -6*E[i]*I[i]/L[i]**2
    C[2*i+2,2*i+2] += 12*E[i]*I[i]/L[i]**3
    C[2*i+2,2*i+3] += -6*E[i]*I[i]/L[i]**2

    C[2*i+3,2*i] += 6*E[i]*I[i]/L[i]**2
    C[2*i+3,2*i+1] += 2*E[i]*I[i]/L[i]
    C[2*i+3,2*i+2] += -6*E[i]*I[i]/L[i]**2
    C[2*i+3,2*i+3] += 4*E[i]*I[i]/L[i]

    M[2*i,2*i] += 13/35*m[i] + m0[i] - alpha[i]
    M[2*i,2*i+1] += 11/210*m[i]*L[i]
    M[2*i,2*i+2] += 9/70*m[i]
    M[2*i,2*i+3] += -13/420*m[i]*L[i]

    M[2*i+1,2*i] += 11/210*m[i]*L[i]
    M[2*i+1,2*i+1] += 1/105*m[i]*L[i]**2 + 1/16*m[i]*(D[i]**2 + d[i]**2) -
1/12*m[i]*L[i]**2 + I_m[i]
    M[2*i+1,2*i+2] += 13/420*m[i]*L[i]
    M[2*i+1,2*i+3] += -1/140*m[i]*L[i]**2

    M[2*i+2,2*i] += 9/70*m[i]
    M[2*i+2,2*i+1] += 13/420*m[i]*L[i]
    M[2*i+2,2*i+2] += 13/35*m[i]
    M[2*i+2,2*i+3] += -11/210*m[i]*L[i]

    M[2*i+3,2*i] += -13/420*m[i]*L[i]
    M[2*i+3,2*i+1] += -1/140*m[i]*L[i]**2
    M[2*i+3,2*i+2] += -11/210*m[i]*L[i]
    M[2*i+3,2*i+3] += 1/105*m[i]*L[i]**2
```

Ця ділянка коду виконує розрахунок елементів матриць «C» та «M» які містять інформацію про властивості ротора. Кількість ітерацій циклу for залежить від кількості ділянок на роторі.

Кожна із ітерацій виконує розрахунок на основі формул, які залежать від значень параметрів «E», «I», «L», «m» та «alpha».

Цей розрахунок дозволяє далі обчислити власні значення та вектори матриці системи.

```
M_inv = np.linalg.inv(M)

Dyn = M_inv.dot(C)
w2, vect2 = np.linalg.eig(Dyn)

descending = w2.argsort()[::-1]
w2 = w2[descending]
vect2 = vect2[:,descending]

w2p = w2[w2 >= 0]
vect2p = vect2[w2 >= 0]
vect = vect2p[:,::2]
w = np.sqrt(w2p)
```

Після створення масивів I (момент інерції) та m (маса) потрібно розрахувати їх значення за допомогою циклу for для кожної ділянки ротору.

Першим кроком за допомогою функції «np.linalg.inv()» проводиться обчислення оберненої матриці M_inv до матриці M

Розрахунок матриці Dyn забезпечується функцією векторного добутку «dot()» матриць «M_inv» та «C»

Для розрахунку власних значень на векторів («w2» та «vect2» відповідно) для матриці «Dyn» використовується функція «np.linalg.eig()»

Наступні три рядки відповідають за обробку значень матриці «w2» та «vect2» сортуванням із значень у порядку спадання значень «w2»

До створених змінних «w2p» «vect2p» записуються невід'ємні значення відповідних матриць із подальшим пошуком кореня із значень матриці «w2p» до змінної «w» яка відповідає за власні частоти коливання ротора.

3.3 Відображення геометрії ротора

```
coords = np.append([0],L)

X = np.zeros(n+1)
for i in range(n+1):
    sum = 0
    for j in range(i):
        sum += L[j]
    X[i] = sum

length = X[-1]
```

Першим кроком потрібно створити змінну, яка буде містити значення координат під назвою «coords». Розрахунок координат кінця ділянки буде проводитися за допомогою циклу for.

```
def ends(num):
    if num == 1:
        return 'st'
    elif num == 2:
        return 'nd'
    elif num == 3:
        return 'rd'
    else:
        return 'th'
```

Функція для визначення закінчень назв частот за порядком. Функція є виключно косметичною.

```
def marks(num):
    if num%5 == 0:
        return 'o'
    elif num%5 == 1:
        return '^'
    elif num%5 == 2:
        return 's'
    elif num%5 == 3:
        return 'D'
    else:
        return 'x'
```

Функція для визначення форми маркеру на графіку. Слугує для полегшення читаємості.

```
plt.subplot(2,1,1)
plt.title('Rotor design')
plt.xlabel('Length, m')
plt.ylabel('Y-axis, m')

GDX = [0]
GDY = [0]
GdX = [0]
GdY = [0]
for i in range(n):
    GDX = np.append(GDX,[X[i],X[i+1],X[i+1]])
    GDY = np.append(GDY,[D[i],D[i],0])
    GdX = np.append(GdX,[X[i],X[i+1],X[i+1]])
    GdY = np.append(GdY,[d[i],d[i],0])

GmX = []
GmY = []
for i in range(n):
    if m0[i] != 0:
        GmX = np.append(GmX,X[i])
        GmY = np.append(GmY,0)

GIX = []
GIY = []
for i in range(n):
    if I_m[i] != 0:
        GIX = np.append(GIX,X[i])
        GIY = np.append(GIY,0)

GkX = []
GkY = []
for i in range(n):
    if k[i] != 0:
        GkX = np.append(GkX,X[i])
        GkY = np.append(GkY,-D[i])

GaX = []
GaY = []
for i in range(n):
    if alpha[i] != 0:
        GaX = np.append(GaX,X[i])
        GaY = np.append(GaY,-D[i])

def not_zero_d():
    sum = 0
    for i in range(n):
        sum += d[i]
    return sum != 0

plt.plot(GDX,GDY,color='gray')
plt.plot(GDX,-GDY,color='gray')
```

```
if not_zero_d():
    plt.plot(GdX,GdY,color='gray',linestyle='--')
    plt.plot(GdX,-GdY,color='gray',linestyle='--')
plt.plot([-0.02*length,1.02*length],[0,0],color='orange',linestyle='-.')
plt.scatter(GmX,GmY,color='red',marker='o',label='Added mass')
plt.scatter(GIX,GIY,color='orange',marker='s',label='Added moment of inertia')
plt.scatter(GkX,GkY,color='blue',marker='^',label='Support')
plt.scatter(GaX,GaY,color='green',marker='^',label='Variable stiffness')
plt.legend()

plt.subplots_adjust(hspace = 0.5)

plt.subplot(2,1,2)
plt.title('Critical frequencies and mode shapes')
plt.xlabel('Length, m')
plt.ylabel('Dimensionless displacement')
plt.grid(color='lightgray',linestyle='-')

x = np.linspace(0,length,n*5)

for i in range(num):
    plt.scatter(X,Y[:,i],marker = marks(i))
    plt.plot(x,f_interp(x,X,Y[:,i]),label=f'{i+1}{ends(i+1)}                mode
shape:\nf$_{i+1}$ = {round(w[-1-i]/(2*pi),1)} Hz')

plt.legend()

plt.show()
```

Ця частина коду виконує дії:

- Для побудови графіку:

- Використовуються координати точок для побудови профілю ротора.

- Графік будується за допомогою методу plt.plot() для лінійних сегментів і методу plt.scatter() для позначення окремих точок.

- Для різних елементів ротора використовуються різні кольори та маркери.

- Налаштування осей та масштабу графіку.

- Побудова графіку критичних частот і форм мод:

- Використовується метод plt.subplot() для створення другого підграфіку.
- Задаються підписи осей та сітка для кращої візуалізації.

- Зображуються критичні частоти (власні значення) та форми коливань (власні вектори).

Метод plt.show() відображає графіки.

3.4 Програмний код розрахунку власних частот ротора

Нижче продемонстровано елементи програмного коду реалізації програми розрахунку на основі мови програмування Python.

```
'''IMPORT MODULES'''


import pathlib, os.path, pandas as pd, numpy as np, math
from math import pi
from scipy import interpolate
import matplotlib.pyplot as plt


'''READ FILE'''


def getfile():
    while True:
        file = input('\nEnter file name *.xlsx: ')
        filename = file + '.xlsx'
        path = pathlib.Path(filename)
```

```
        if path.exists():
            return path
        else:
            print('Incorrect file name.')


file = getfile()


'''NUMBER OF EIGENFREQUENCIES'''


def getnumber():
    while True:
        number = input("\nEnter a number of frequencies: ")
        if number.isdigit():
            return number
        else:
            print('Incorrect number.')


num = int(getnumber())
```

```
'''READ DATA'''

A = pd.read_excel(file)
A = np.asarray(A)
A = A[:,1:]

'''CALCULATIONS'''

n, cols = A.shape
C = np.zeros((2*(n+1),2*(n+1)))
M = np.zeros((2*(n+1),2*(n+1)))
L, D, d, m0, I_m, k, alpha, dens, E = A[:,0], A[:,1], A[:,2], A[:,3], A[:,4], A[:,5], A[:,6], A[:,7], A[:,8]

I, m = [], []
for i in range(n):
    I.append(pi*(D[i]**4-d[i]**4)/64)
    m.append(dens[i]*L[i]*pi*(D[i]**2-d[i]**2)/4)
```

```
for i in range(n):
    C[2*i,2*i] += 12*E[i]*I[i]/L[i]**3 + k[i]
    C[2*i,2*i+1] += 6*E[i]*I[i]/L[i]**2
    C[2*i,2*i+2] += -12*E[i]*I[i]/L[i]**3
    C[2*i,2*i+3] += 6*E[i]*I[i]/L[i]**2

    C[2*i+1,2*i] += 6*E[i]*I[i]/L[i]**2
    C[2*i+1,2*i+1] += 4*E[i]*I[i]/L[i]
    C[2*i+1,2*i+2] += -6*E[i]*I[i]/L[i]**2
    C[2*i+1,2*i+3] += 2*E[i]*I[i]/L[i]

    C[2*i+2,2*i] += -12*E[i]*I[i]/L[i]**3
    C[2*i+2,2*i+1] += -6*E[i]*I[i]/L[i]**2
    C[2*i+2,2*i+2] += 12*E[i]*I[i]/L[i]**3
    C[2*i+2,2*i+3] += -6*E[i]*I[i]/L[i]**2

    C[2*i+3,2*i] += 6*E[i]*I[i]/L[i]**2
    C[2*i+3,2*i+1] += 2*E[i]*I[i]/L[i]
    C[2*i+3,2*i+2] += -6*E[i]*I[i]/L[i]**2
    C[2*i+3,2*i+3] += 4*E[i]*I[i]/L[i]
```

```
M[2*i,2*i] += 13/35*m[i] + m0[i] - alpha[i]

M[2*i,2*i+1] += 11/210*m[i]*L[i]

M[2*i,2*i+2] += 9/70*m[i]

M[2*i,2*i+3] += -13/420*m[i]*L[i]


M[2*i+1,2*i] += 11/210*m[i]*L[i]

M[2*i+1,2*i+1] += 1/105*m[i]*L[i]**2 + 1/16*m[i]*(D[i]**2 + d[i]**2)
- 1/12*m[i]*L[i]**2 + I_m[i]

M[2*i+1,2*i+2] += 13/420*m[i]*L[i]

M[2*i+1,2*i+3] += -1/140*m[i]*L[i]**2


M[2*i+2,2*i] += 9/70*m[i]

M[2*i+2,2*i+1] += 13/420*m[i]*L[i]

M[2*i+2,2*i+2] += 13/35*m[i]

M[2*i+2,2*i+3] += -11/210*m[i]*L[i]


M[2*i+3,2*i] += -13/420*m[i]*L[i]

M[2*i+3,2*i+1] += -1/140*m[i]*L[i]**2

M[2*i+3,2*i+2] += -11/210*m[i]*L[i]
```

```
    M[2*i+3,2*i+3] += 1/105*m[i]*L[i]**2


M_inv = np.linalg.inv(M)

Dyn = M_inv.dot(C)

w2, vect2 = np.linalg.eig(Dyn)


descending = w2.argsort()[::-1]

w2 = w2[descending]

vect2 = vect2[:,descending]


w2p = w2[w2 >= 0]

vect2p = vect2[w2 >= 0]

vect = vect2p[:,::2]

w = np.sqrt(w2p)


'''PRINT RESULTS'''


num = min(num,len(w))

br = '----------------------'
```

```
print('\n'+br+'\nEIGENFREQUENCIES, Hz: \n'+br)
for i in range(num):
    print(f'f[{i+1}] = {round(w[-1-i]/(2*pi),1)}')

CC = C[0:2*n+1,0:2*n+1]
MM = M[0:2*n+1,0:2*n+1]

CC0 = np.zeros(2*n+1)
MM0 = np.zeros(2*n+1)
for i in range(2*n+1):
    CC0[i] = C[i,2*n+1]
    MM0[i] = M[i,2*n+1]

def DD(num):
    return CC - w[-1-num]**2*MM

def DDD(num):
    return CC0 - w[-1-num]**2*MM0

V = np.zeros((2*n+1,num))
```

```
for i in range(num):
    V[:,i] = -np.linalg.inv(DD(i)).dot(DDD(i))

Y = V[::2,:]

for i in range(num):
    Y[:,i] = Y[:,i]/max(max(Y[:,i]),max(-Y[:,i]))

print('\n'+br+'\nMODE SHAPES: ')
for i in range(num):
    np.set_printoptions(precision=3)
    print(br+f'\nmode_shape[{i+1}] = {Y[:,i]}')

print('\n'+br+'\nCompleted successfully.')

'''VISUALIZATION'''

coords = np.append([0],L)
```

```
X = np.zeros(n+1)
for i in range(n+1):
    sum = 0
    for j in range(i):
        sum += L[j]
    X[i] = sum

length = X[-1]

def f_interp(x,X,Y):
    y = interpolate.splrep(X,Y)
    return interpolate.splev(x,y)

def ends(num):
    if num == 1:
        return 'st'
    elif num == 2:
        return 'nd'
    elif num == 3:
        return 'rd'
```

```
    else:
        return 'th'


def marks(num):
    if num%5 == 0:
        return 'o'
    elif num%5 == 1:
        return '^'
    elif num%5 == 2:
        return 's'
    elif num%5 == 3:
        return 'D'
    else:
        return 'x'


plt.gcf().canvas.manager.set_window_title(f'(C) Pavlenko I., Sumy State University, 2022 | "Critical frequencies of the rotor" | File: {file}')


plt.subplot(2,1,1)
plt.title('Rotor design')
```

```
plt.xlabel('Length, m')
plt.ylabel('Y-axis, m')

GDX = [0]
GDY = [0]
GdX = [0]
GdY = [0]
for i in range(n):
    GDX = np.append(GDX,[X[i],X[i+1],X[i+1]])
    GDY = np.append(GDY,[D[i],D[i],0])
    GdX = np.append(GdX,[X[i],X[i+1],X[i+1]])
    GdY = np.append(GdY,[d[i],d[i],0])

GmX = []
GmY = []
for i in range(n):
    if m0[i] != 0:
        GmX = np.append(GmX,X[i])
        GmY = np.append(GmY,0)
```

```
GIX = []
GIY = []
for i in range(n):
    if I_m[i] != 0:
        GIX = np.append(GIX,X[i])
        GIY = np.append(GIY,0)

GkX = []
GkY = []
for i in range(n):
    if k[i] != 0:
        GkX = np.append(GkX,X[i])
        GkY = np.append(GkY,-D[i])

GaX = []
GaY = []
for i in range(n):
    if alpha[i] != 0:
        GaX = np.append(GaX,X[i])
        GaY = np.append(GaY,-D[i])
```

```
    def not_zero_d():
        sum = 0
        for i in range(n):
            sum += d[i]
        return sum != 0


    plt.plot(GDX,GDY,color='gray')
    plt.plot(GDX,-GDY,color='gray')
    if not_zero_d():
        plt.plot(GdX,GdY,color='gray',linestyle='--')
        plt.plot(GdX,-GdY,color='gray',linestyle='--')
    plt.plot([-0.02*length,1.02*length],[0,0],color='orange',linestyle='-.')
    plt.scatter(GmX,GmY,color='red',marker='o',label='Added mass')
    plt.scatter(GIX,GIY,color='orange',marker='s',label='Added moment of
inertia')
    plt.scatter(GkX,GkY,color='blue',marker='^',label='Support')
    plt.scatter(GaX,GaY,color='green',marker='^',label='Variable stiffness')
    plt.legend()
```

```
plt.subplots_adjust(hspace = 0.5)


plt.subplot(2,1,2)
plt.title('Critical frequencies and mode shapes')
plt.xlabel('Length, m')
plt.ylabel('Dimensionless displacement')
plt.grid(color='lightgray',linestyle='-')


x = np.linspace(0,length,n*5)


for i in range(num):
    plt.scatter(X,Y[:,i],marker = marks(i))
    plt.plot(x,f_interp(x,X,Y[:,i]),label=f'{i+1}{ends(i+1)} mode shape:\nf$_{i+1}$ = {round(w[-1-i]/(2*pi),1)} Hz')


plt.legend()


plt.show()
```

3.5 Створення графічного інтерфейсу програми

Графічний інтерфейс програми формується за допомогою наступного програмного коду:

```
from PyQt6 import QtCore, QtGui, QtWidgets


# Main window design:
class Ui_MainWindow(object):
    def setupUi(self, MainWindow):
        MainWindow.setObjectName("MainWindow")
        MainWindow.resize(800, 500)
        icon0 = QtGui.QIcon()###
        icon0.addPixmap(QtGui.QPixmap("images/icon.png"),
QtGui.QIcon.Mode.Normal, QtGui.QIcon.State.Off)###
        MainWindow.setWindowIcon(icon0)###
        self.centralwidget = QtWidgets.QWidget(MainWindow)
        self.centralwidget.setObjectName("centralwidget")
        self.horizontalLayout = QtWidgets.QHBoxLayout(self.centralwidget)
        self.horizontalLayout.setObjectName("horizontalLayout")
        self.textEdit = QtWidgets.QTextEdit(self.centralwidget)
        self.textEdit.setObjectName("textEdit")
```

```
self.horizontalLayout.addWidget(self.textEdit)
MainWindow.setCentralWidget(self.centralwidget)
self.menubar = QtWidgets.QMenuBar(MainWindow)
self.menubar.setGeometry(QtCore.QRect(0, 0, 800, 22))
self.menubar.setObjectName("menubar")
self.menuFile = QtWidgets.QMenu(self.menubar)
self.menuFile.setObjectName("menuFile")
self.menuEdit = QtWidgets.QMenu(self.menubar)
self.menuEdit.setObjectName("menuEdit")
self.menuFormat = QtWidgets.QMenu(self.menubar)
self.menuFormat.setObjectName("menuFormat")
self.menuAbout = QtWidgets.QMenu(self.menubar)
self.menuAbout.setObjectName("menuAbout")
MainWindow.setMenuBar(self.menubar)
self.statusbar = QtWidgets.QStatusBar(MainWindow)
self.statusbar.setObjectName("statusbar")
MainWindow.setStatusBar(self.statusbar)
self.toolBar = QtWidgets.QToolBar(MainWindow)
self.toolBar.setObjectName("toolBar")
```

```
        MainWindow.addToolBar(QtCore.Qt.ToolBarArea.TopToolBarArea,
self.toolBar)
        self.actionNew = QtGui.QAction(MainWindow)
        icon = QtGui.QIcon()
        icon.addPixmap(QtGui.QPixmap("images/new.jpg"),
QtGui.QIcon.Mode.Normal, QtGui.QIcon.State.Off)
        self.actionNew.setIcon(icon)
        self.actionNew.setObjectName("actionNew")

        self.actionOpen = QtGui.QAction(MainWindow)
        icon1 = QtGui.QIcon()
        icon1.addPixmap(QtGui.QPixmap("images/open.jpg"),
QtGui.QIcon.Mode.Normal, QtGui.QIcon.State.Off)
        self.actionOpen.setIcon(icon1)
        self.actionOpen.setObjectName("actionOpen")

        self.actionCalculate = QtGui.QAction(MainWindow)
        icon_calculate = QtGui.QIcon()
        icon_calculate.addPixmap(QtGui.QPixmap("images/calculate.jpg"),
QtGui.QIcon.Mode.Normal, QtGui.QIcon.State.Off)
        self.actionCalculate.setIcon(icon_calculate)
```

```
        self.actionCalculate.setObjectName("actionCalculate")


        self.actionSave = QtGui.QAction(MainWindow)
        icon2 = QtGui.QIcon()
        icon2.addPixmap(QtGui.QPixmap("images/save.png"),
QtGui.QIcon.Mode.Normal, QtGui.QIcon.State.Off)
        self.actionSave.setIcon(icon2)
        self.actionSave.setObjectName("actionSave")


        self.actionOpenTxt = QtGui.QAction(MainWindow)
        icon1txt = QtGui.QIcon()
        icon1txt.addPixmap(QtGui.QPixmap("images/open_txt.png"),
QtGui.QIcon.Mode.Normal, QtGui.QIcon.State.Off)
        self.actionOpenTxt.setIcon(icon1txt)
        self.actionOpenTxt.setObjectName("actionOpenTxt")


        self.actionPrint = QtGui.QAction(MainWindow)
        icon3 = QtGui.QIcon()
        icon3.addPixmap(QtGui.QPixmap("images/print.png"),
QtGui.QIcon.Mode.Normal, QtGui.QIcon.State.Off)
        self.actionPrint.setIcon(icon3)
```

```
        self.actionPrint.setObjectName("actionPrint")
        self.actionPrint_Preview = QtGui.QAction(MainWindow)
        icon4 = QtGui.QIcon()
        icon4.addPixmap(QtGui.QPixmap("images/print_preview.png"),
QtGui.QIcon.Mode.Normal, QtGui.QIcon.State.Off)
        self.actionPrint_Preview.setIcon(icon4)
        self.actionPrint_Preview.setObjectName("actionPrint_Preview")
        self.actionExport_PDF = QtGui.QAction(MainWindow)
        icon5 = QtGui.QIcon()
        icon5.addPixmap(QtGui.QPixmap("images/export_pdf.png"),
QtGui.QIcon.Mode.Normal, QtGui.QIcon.State.Off)
        self.actionExport_PDF.setIcon(icon5)
        self.actionExport_PDF.setObjectName("actionExport_PDF")
        self.actionQuit = QtGui.QAction(MainWindow)
        icon6 = QtGui.QIcon()
        icon6.addPixmap(QtGui.QPixmap("images/quit.jfif"),
QtGui.QIcon.Mode.Normal, QtGui.QIcon.State.Off)
        self.actionQuit.setIcon(icon6)
        self.actionQuit.setObjectName("actionQuit")
        self.actionUndo = QtGui.QAction(MainWindow)
        icon7 = QtGui.QIcon()
```

```
        icon7.addPixmap(QtGui.QPixmap("images/undo.png"),
QtGui.QIcon.Mode.Normal, QtGui.QIcon.State.Off)
        self.actionUndo.setIcon(icon7)
        self.actionUndo.setObjectName("actionUndo")
        self.actionRedo = QtGui.QAction(MainWindow)
        icon8 = QtGui.QIcon()
        icon8.addPixmap(QtGui.QPixmap("images/redo.png"),
QtGui.QIcon.Mode.Normal, QtGui.QIcon.State.Off)
        self.actionRedo.setIcon(icon8)
        self.actionRedo.setObjectName("actionRedo")
        self.actionCut = QtGui.QAction(MainWindow)
        icon9 = QtGui.QIcon()
        icon9.addPixmap(QtGui.QPixmap("images/cut.jfif"),
QtGui.QIcon.Mode.Normal, QtGui.QIcon.State.Off)
        self.actionCut.setIcon(icon9)
        self.actionCut.setObjectName("actionCut")
        self.actionCopy = QtGui.QAction(MainWindow)
        icon10 = QtGui.QIcon()
        icon10.addPixmap(QtGui.QPixmap("images/copy.jfif"),
QtGui.QIcon.Mode.Normal, QtGui.QIcon.State.Off)
        self.actionCopy.setIcon(icon10)
```

```
        self.actionCopy.setObjectName("actionCopy")
        self.actionPaste = QtGui.QAction(MainWindow)
        icon11 = QtGui.QIcon()
        icon11.addPixmap(QtGui.QPixmap("images/paste.png"),
QtGui.QIcon.Mode.Normal, QtGui.QIcon.State.Off)
        self.actionPaste.setIcon(icon11)
        self.actionPaste.setObjectName("actionPaste")
        self.actionBold = QtGui.QAction(MainWindow)
        icon12 = QtGui.QIcon()
        icon12.addPixmap(QtGui.QPixmap("images/bold.png"),
QtGui.QIcon.Mode.Normal, QtGui.QIcon.State.Off)
        self.actionBold.setIcon(icon12)
        self.actionBold.setObjectName("actionBold")
        self.actionItalic = QtGui.QAction(MainWindow)
        icon13 = QtGui.QIcon()
        icon13.addPixmap(QtGui.QPixmap("images/italic.jfif"),
QtGui.QIcon.Mode.Normal, QtGui.QIcon.State.Off)
        self.actionItalic.setIcon(icon13)
        self.actionItalic.setObjectName("actionItalic")
        self.actionUnderline = QtGui.QAction(MainWindow)
        icon14 = QtGui.QIcon()
```

```
            icon14.addPixmap(QtGui.QPixmap("images/underline.png"),
QtGui.QIcon.Mode.Normal, QtGui.QIcon.State.Off)
            self.actionUnderline.setIcon(icon14)
            self.actionUnderline.setObjectName("actionUnderline")
            self.actionLeft = QtGui.QAction(MainWindow)
            icon15 = QtGui.QIcon()
            icon15.addPixmap(QtGui.QPixmap("images/left.png"),
QtGui.QIcon.Mode.Normal, QtGui.QIcon.State.Off)
            self.actionLeft.setIcon(icon15)
            self.actionLeft.setObjectName("actionLeft")
            self.actionRight = QtGui.QAction(MainWindow)
            icon16 = QtGui.QIcon()
            icon16.addPixmap(QtGui.QPixmap("images/right.png"),
QtGui.QIcon.Mode.Normal, QtGui.QIcon.State.Off)
            self.actionRight.setIcon(icon16)
            self.actionRight.setObjectName("actionRight")
            self.actionCenter = QtGui.QAction(MainWindow)
            icon17 = QtGui.QIcon()
            icon17.addPixmap(QtGui.QPixmap("images/center.png"),
QtGui.QIcon.Mode.Normal, QtGui.QIcon.State.Off)
            self.actionCenter.setIcon(icon17)
```

```
        self.actionCenter.setObjectName("actionCenter")
        self.actionJustify = QtGui.QAction(MainWindow)
        icon18 = QtGui.QIcon()
        icon18.addPixmap(QtGui.QPixmap("images/justify.png"),
QtGui.QIcon.Mode.Normal, QtGui.QIcon.State.Off)
        self.actionJustify.setIcon(icon18)
        self.actionJustify.setObjectName("actionJustify")
        self.actionFont = QtGui.QAction(MainWindow)
        icon19 = QtGui.QIcon()
        icon19.addPixmap(QtGui.QPixmap("images/font.jfif"),
QtGui.QIcon.Mode.Normal, QtGui.QIcon.State.Off)
        self.actionFont.setIcon(icon19)
        self.actionFont.setObjectName("actionFont")
        self.actionColor = QtGui.QAction(MainWindow)
        icon20 = QtGui.QIcon()
        icon20.addPixmap(QtGui.QPixmap("images/color.png"),
QtGui.QIcon.Mode.Normal, QtGui.QIcon.State.Off)
        self.actionColor.setIcon(icon20)
        self.actionColor.setObjectName("actionColor")
        self.actionAbout_App = QtGui.QAction(MainWindow)
        icon21 = QtGui.QIcon()
```

```
        icon21.addPixmap(QtGui.QPixmap("images/about_app.jfif"),
QtGui.QIcon.Mode.Normal, QtGui.QIcon.State.Off)
        self.actionAbout_App.setIcon(icon21)
        self.actionAbout_App.setObjectName("actionAbout_App")
        self.actionHelp = QtGui.QAction(MainWindow)
        icon22 = QtGui.QIcon()
        icon22.addPixmap(QtGui.QPixmap("images/help.jfif"),
QtGui.QIcon.Mode.Normal, QtGui.QIcon.State.Off)
        self.actionHelp.setIcon(icon22)
        self.actionHelp.setObjectName("actionHelp")
        self.menuFile.addAction(self.actionNew)
        self.menuFile.addAction(self.actionOpen)
        self.menuFile.addAction(self.actionCalculate)
        self.menuFile.addAction(self.actionSave)
        self.menuFile.addAction(self.actionOpenTxt)
        self.menuFile.addSeparator()
        self.menuFile.addAction(self.actionPrint)
        self.menuFile.addAction(self.actionPrint_Preview)
        self.menuFile.addAction(self.actionExport_PDF)
        self.menuFile.addSeparator()
        self.menuFile.addAction(self.actionQuit)
```

```
self.menuEdit.addAction(self.actionUndo)
self.menuEdit.addAction(self.actionRedo)
self.menuEdit.addSeparator()
self.menuEdit.addAction(self.actionCut)
self.menuEdit.addAction(self.actionCopy)
self.menuEdit.addAction(self.actionPaste)
self.menuFormat.addAction(self.actionBold)
self.menuFormat.addAction(self.actionItalic)
self.menuFormat.addAction(self.actionUnderline)
self.menuFormat.addSeparator()
self.menuFormat.addAction(self.actionLeft)
self.menuFormat.addAction(self.actionRight)
self.menuFormat.addAction(self.actionCenter)
self.menuFormat.addAction(self.actionJustify)
self.menuFormat.addSeparator()
self.menuFormat.addAction(self.actionFont)
self.menuFormat.addAction(self.actionColor)
self.menuAbout.addAction(self.actionAbout_App)
self.menuAbout.addAction(self.actionHelp)
self.menubar.addAction(self.menuFile.menuAction())
```

```
self.menubar.addAction(self.menuEdit.menuAction())
self.menubar.addAction(self.menuFormat.menuAction())
self.menubar.addAction(self.menuAbout.menuAction())
self.toolBar.addAction(self.actionNew)
self.toolBar.addAction(self.actionOpen)
self.toolBar.addAction(self.actionCalculate)
self.toolBar.addAction(self.actionSave)
self.toolBar.addAction(self.actionOpenTxt)
self.toolBar.addAction(self.actionPrint)
self.toolBar.addAction(self.actionPrint_Preview)
self.toolBar.addAction(self.actionExport_PDF)
self.toolBar.addAction(self.actionQuit)
self.toolBar.addAction(self.actionUndo)
self.toolBar.addAction(self.actionRedo)
self.toolBar.addAction(self.actionCut)
self.toolBar.addAction(self.actionCopy)
self.toolBar.addAction(self.actionPaste)
self.toolBar.addAction(self.actionBold)
self.toolBar.addAction(self.actionItalic)
self.toolBar.addAction(self.actionUnderline)
```

```
        self.toolBar.addAction(self.actionLeft)
        self.toolBar.addAction(self.actionRight)
        self.toolBar.addAction(self.actionCenter)
        self.toolBar.addAction(self.actionJustify)
        self.toolBar.addAction(self.actionFont)
        self.toolBar.addAction(self.actionColor)
        self.toolBar.addAction(self.actionAbout_App)
        self.toolBar.addAction(self.actionHelp)

        self.retranslateUi(MainWindow)
        QtCore.QMetaObject.connectSlotsByName(MainWindow)

    def retranslateUi(self, MainWindow):
        _translate = QtCore.QCoreApplication.translate
        MainWindow.setWindowTitle(_translate("MainWindow", "Critical
Frequencies App"))
        self.menuFile.setTitle(_translate("MainWindow", "File"))
        self.menuEdit.setTitle(_translate("MainWindow", "Edit"))
        self.menuFormat.setTitle(_translate("MainWindow", "Format"))
        self.menuAbout.setTitle(_translate("MainWindow", "About"))
```

```
        self.toolBar.setWindowTitle(_translate("MainWindow", "toolBar"))
        self.actionNew.setText(_translate("MainWindow", "New"))
        self.actionNew.setShortcut(_translate("MainWindow", "Ctrl+N"))
        self.actionOpen.setText(_translate("MainWindow", "Open and Run"))
        self.actionOpen.setShortcut(_translate("MainWindow", "Ctrl+O"))
        self.actionCalculate.setText(_translate("MainWindow", "Calculate"))
        self.actionCalculate.setShortcut(_translate("MainWindow", "Alt+C"))
        self.actionSave.setText(_translate("MainWindow", "Save results"))
        self.actionSave.setShortcut(_translate("MainWindow", "Ctrl+S"))
        self.actionOpenTxt.setText(_translate("MainWindow", "Open results"))
        self.actionOpenTxt.setShortcut(_translate("MainWindow",
"Ctrl+Shift+O"))
        self.actionPrint.setText(_translate("MainWindow", "Print results"))
        self.actionPrint.setShortcut(_translate("MainWindow", "Ctrl+P"))
        self.actionPrint_Preview.setText(_translate("MainWindow",      "Print
preview"))
        self.actionPrint_Preview.setShortcut(_translate("MainWindow",
"Ctrl+Shift+P"))
        self.actionExport_PDF.setText(_translate("MainWindow",      "Export
PDF"))
```

```
        self.actionExport_PDF.setShortcut(_translate("MainWindow",
"Ctrl+E"))
        self.actionQuit.setText(_translate("MainWindow", "Quit"))
        self.actionQuit.setShortcut(_translate("MainWindow", "Ctrl+Q"))
        self.actionUndo.setText(_translate("MainWindow", "Undo"))
        self.actionUndo.setShortcut(_translate("MainWindow", "Ctrl+Z"))
        self.actionRedo.setText(_translate("MainWindow", "Redo"))
        self.actionRedo.setShortcut(_translate("MainWindow", "Ctrl+Y"))
        self.actionCut.setText(_translate("MainWindow", "Cut"))
        self.actionCut.setShortcut(_translate("MainWindow", "Ctrl+X"))
        self.actionCopy.setText(_translate("MainWindow", "Copy"))
        self.actionCopy.setShortcut(_translate("MainWindow", "Ctrl+C"))
        self.actionPaste.setText(_translate("MainWindow", "Paste"))
        self.actionPaste.setShortcut(_translate("MainWindow", "Ctrl+V"))
        self.actionBold.setText(_translate("MainWindow", "Bold"))
        self.actionBold.setShortcut(_translate("MainWindow", "Ctrl+B"))
        self.actionItalic.setText(_translate("MainWindow", "Italic"))
        self.actionItalic.setShortcut(_translate("MainWindow", "Ctrl+I"))
        self.actionUnderline.setText(_translate("MainWindow", "Underline"))
        self.actionUnderline.setShortcut(_translate("MainWindow", "Ctrl+U"))
```

```
            self.actionLeft.setText(_translate("MainWindow", "Left"))
            self.actionLeft.setShortcut(_translate("MainWindow", "Ctrl+L"))
            self.actionRight.setText(_translate("MainWindow", "Right"))
            self.actionRight.setShortcut(_translate("MainWindow", "Ctrl+R"))
            self.actionCenter.setText(_translate("MainWindow", "Center"))
            self.actionCenter.setShortcut(_translate("MainWindow", "Ctrl+E"))
            self.actionJustify.setText(_translate("MainWindow", "Justify"))
            self.actionJustify.setShortcut(_translate("MainWindow", "Ctrl+J"))
            self.actionFont.setText(_translate("MainWindow", "Font"))
            self.actionFont.setShortcut(_translate("MainWindow", "Ctrl+Shift+F"))
            self.actionColor.setText(_translate("MainWindow", "Color"))
            self.actionColor.setShortcut(_translate("MainWindow",
"Ctrl+Shift+C"))
            self.actionAbout_App.setText(_translate("MainWindow",          "About
App"))
            self.actionAbout_App.setShortcut(_translate("MainWindow",
"Ctrl+Shift+A"))
            self.actionHelp.setText(_translate("MainWindow", "Help"))
            self.actionHelp.setShortcut(_translate("MainWindow", "Ctrl+H"))
```

3.6 Висновки до третього розділу

У розділі "Побудова програми" була описана реалізація комп'ютерної програми для розрахунку власних частот ротора на основі мови програмування Python. Програма має на меті вивчення вібраційного стану ротора і забезпечення виводу результатів у вигляді графіку.

У розділі були представлені основні етапи розробки програми. Починаючи з вхідних даних, які отримуються з файлу у форматі *.xlsx, була використана бібліотека Pandas для зчитування та обробки даних.

З використанням бібліотеки Matplotlib була забезпечена візуалізація результатів у вигляді графіку, що дозволяє зрозуміти характеристики вібраційного стану ротора.

У результаті реалізації програми було досягнуто поставленої мети з аналізу вібраційного стану ротора та виведення результатів у зручному для аналізу графічному вигляді. Побудована програма є потужним інструментом для вивчення та аналізу динаміки роторів, що може бути використана у різних інженерних дисциплінах.

4 ДІАГНОСТИКА ВІБРАЦІЙНОГО СТАНУ РОТОРА

4.1 Розрахунок за розробленою програмою

Вхідні дані

Об'єктом дослідження є вільні та вимушені коливання гнучкого ротора центрифугального компресора 225GC2-135/12-50M1245 [17] на ТЕЦ Навої потужністю 1643 МВт (Янгі-Арік, Узбекистан) з наступними технічними параметрами: подача 25 $м^3$/с; вхідний тиск 1,20 МПа; вихідний тиск 4,95 МПа; робочою частотою 934 рад/с; номінальною потужністю 7,33 МВт.

Розрахунок власних частот дозволяє визначити потенційно небезпечні швидкості ротора при його обертанні на робочій частоті, яка дорівнює 934 рад/с. Тому для загальних розрахунків була прийнята постійна жорсткість підшипника, рівна $c_w = 2,94 \cdot 108$ Н/м при робочій швидкості ротора.

Етап вхідних даних. На цьому етапі програма очікує від користувача назву файлу із вхідними даними, який оформлено згідно до шаблону, що представлено на Таблиці 4.1, та кількість частот, які треба розрахувати.

Таблиця 4.1 – Шаблон заповнення файлу вхідних даних

Number	L [m]	D [m]	d [m]	m [kg]	I [kg*m^2]	c [N/m]	a [N*s^2/m]	dens [kg/m^3]	E, Pa
1									
2									
3									
…									

Enter file name *.xlsx: 1

Enter a number of frequencies: 3

EIGENFREQUENCIES, Hz:

---

f[1] = 318

f[2] = 1148

f[3] = 1906

MODE SHAPES:

---

mode_shape[1] = [6.422e-01 1.304e-06 -5.180e-01 -1.000e+00 -3.539e-01 -2.656e-12

  5.988e-02 6.691e-02 1.063e-01 5.987e-02 1.257e-02 9.217e-03

  2.455e-12 -1.542e-12 8.202e-03 1.233e-02 2.956e-02]

---

mode_shape[2] = [6.071e-04 -4.738e-09 -1.700e-04 -3.119e-03 -8.802e-03 1.541e-13

  5.158e-03 6.183e-03 1.673e-02 1.962e-02 8.223e-03 6.741e-03

 -5.414e-12 7.091e-12 1.322e-01 2.491e-01 1.000e+00]

---

mode_shape[3] = [-1.000e+00 1.500e-06 4.871e-01 7.719e-01 7.198e-01 3.670e-12

 -1.931e-01 -2.218e-01 -4.346e-01 -2.957e-01 -6.495e-02 -4.764e-02

-1.599e-11 1.015e-11 -8.829e-03 1.336e-03 1.292e-01]

Completed successfully.

4.2 Розрахунок за ANSYS

Програмне забезпечення ANSYS надає можливість моделювання динаміки ротора за допомогою методу скінченних елементів як для двовимірних балок, так і для тривимірних скінченних елементів. Однак, перший варіант не є доцільним, оскільки його результати фундаментально не відрізняються від попередньо обчислених за допомогою Python. [18]

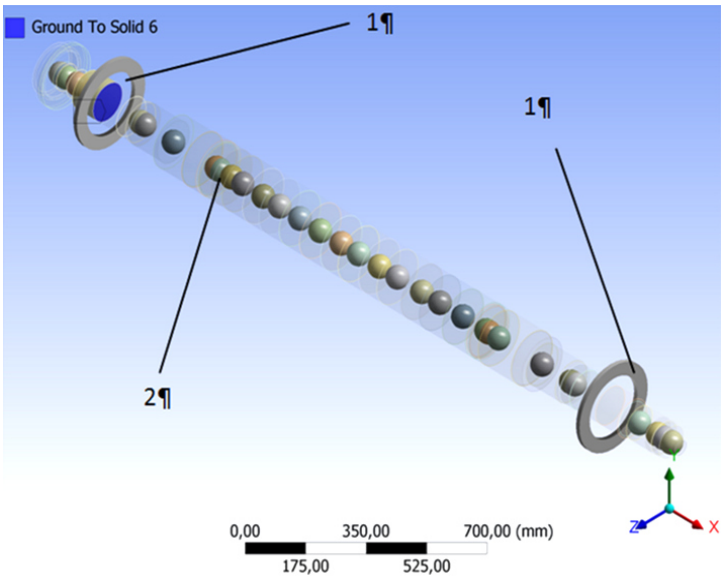

Рисунок 4.1 – Геометрія ротора в Ansys: 1 – підшипники, 2 – еквівалентна маса

Слід зазначити, що можливості програмного забезпечення ANSYS не дозволяють безпосередньо обчислювати критичні частоти, як це можливо з використанням розробленої програми Тому результатом дослідження в ANSYS буде побудова та подальший аналіз діаграми Кемпбелла.

Для розрахованої жорсткості підшипника в діапазоні обертових швидкостей від 0 до 3000 рад/с з кроком 250 рад/с, включаючи робочу швидкість ротора 934 рад/с, були визначені власні частоти. У результаті була побудована діаграма Кемпбелла (Рис. 4.2), де по осі X відображена робоча частота, а по осі Y - власна частота.

На Рис. 4.2 показані результати розрахунку критичних частот ротора. Відповідні числові значення наведені у Таблиці 4.2.

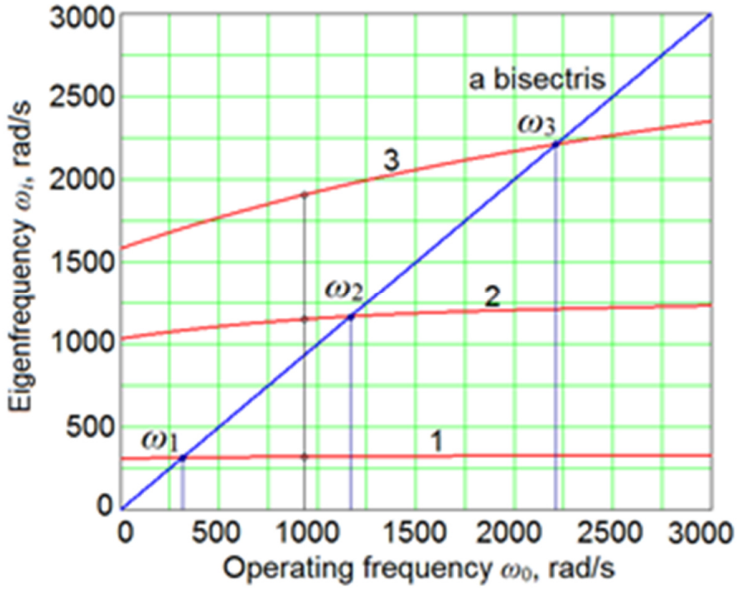

Рисунок 4.2 – Діаграма Кампбелла

Таблиця 4.2 – Власні частоти, рад/с

Номер форми мод	Власні частоти		
	Python	ANSYS	Відносна похибка,%
1	317	319	0.3
2	1148	1125	2.0
3	1906	1867	2.0

ВИСНОВКИ

У результаті проведеної роботи можна зробити висновки, що розроблена програма у сфері дослідження роторної динаміки виявляється досить доцільною, оскільки вона надає гнучкість, швидкість та точність при розрахунках критичних частот ротора, що дозволяє отримувати більш детальні та цінні результати.

У третьому розділі дипломного проекту була проведена розробка програмного забезпечення для визначення власних частот ротора. Для цього було використано математичну модель ротора, яка враховує його геометричні та механічні характеристики, такі як довжина, діаметр, жорсткість підшипників та інші фактори.

Під час розробки програми було використано мову програмування Python та необхідні бібліотеки для обчислень. Програма була налаштована на вхідні дані, такі як параметри ротора та умови роботи, і виводила власні частоти та режими коливань ротора.

Результати роботи програми були порівнянні з модальним аналізом за допомогою програмного забезпечення ANSYS. Виявлено, що програма дає точні та надійні результати, які збігаються з очікуваними значеннями.

Більш того, використання запропонованого підходу уникає необхідності побудови діаграми Кемпбелла та значно зменшить час, необхідний для підготовки та виконання розрахунків. При цьому, цей підхід не впливає на точність результатів розрахунку.

Робота виконувалась у рамках НДР № БФ/26-2021 на Виконання завдань перспективного плану розвитку наукового напряму "Технічні науки" Сумського державного університету (держреєстрація № 0121U112684).

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. https://www.python.org/doc/essays/blurb/
2. https://scipy-lectures.org/intro/intro.html#why-python
3. Adis J. Muminovic, Sanjin Braut, Adil Muminovic, Isad Saric (2014) Numerical and Analytical Analysis of Elastic Rotor Natural Frequency
4. Lukasz Brenkacz, Grzegorz Zywica, Malgorzata Bogulicz (2018) Numerical Analysis of the Rotor of a 30 kW ORC Microturbine Considering Properties of Aerodynamic Gas Bearings
5. Симоновський В. І., Павленко І. В., Калініченко П. М., Дем'яненко М. М., Вашист Б. В., Вербовий А. Є. Дослідження динаміки роторів турбонасосних агрегатів та поршневих компресорних установок. Звіт про науково-дослідну роботу. Суми, 2018
6. Павленко І. В., Симоновський В. І. Методи ідентифікації параметрів математичних моделей коливальних процесів : монографія. Суми : Сумський державний університет, 2020. 145 с.
7. Ankur Saxenaa,* , Anand Pareya , Manoj Chouksey (2016) Study of Modal Characteristics of a geared rotor system
8. Steve J. Rothberg, Ben J. Halkon, Mario Tirabassi, Chris Pusey (2012) Radial vibration measurements directly from rotors using laser vibrometry: The effects of surface roughness, instrument misalignments and pseudo-vibration.
9. ANSYS CFX Tutorials Release 15.0, ANSYS, Inc., 2013.
10. https://www.mathworks.com/academia.html?s_tid=gn_acad
11. https://numpy.org/doc/stable/
12. https://docs.scipy.org/doc/scipy/
13. https://matplotlib.org/stable/index.html#learning-resources
14. https://docs.python.org/3/tutorial/index.html

15. https://pandas.pydata.org/docs/
16. https://www.iso.org/standard/27092.html
17. Pavlenko, I., Simonovskiy, V., Verbovyi, A., Ivchenko, O., Ivanov, V. (2022). Rotor Dynamics and Stability of the Centrifugal Pump CPN 600-35 for Nuclear Power Plants.
18. Pavlenko, I., Neamtu, C., Verbovyi, A., Pitel, J., Ivanov, V., Pop, G. (2019) Using computer modeling and artificial neural networks for ensuring the vibration reliability of rotors. 2nd International Workshop on Computer Modeling and Intelligent Systems. CMIS 2019. CEUR Workshop Proceedings, Vol. 2353, pp. 702-716