Київський національний університет

імені Тараса Шевченка

Факультет комп'ютерних наук та кібернетики

Кафедра обчислювальної математики

Кваліфікаційна робота

на здобуття супеня бакалавра

за спеціальністю 113 Прикладна математика

на тему:

Спектральні та крайові еліптичні задачі на комірках з порожнинами

Виконав студент 4 курсу бакалаврату

Ларіонов Андрій Віталійович _____

Науковий керівник:

професор, доктор фіз.-мат наук

Семенов Володимир Вікторович _____

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

Студент _____ _____

Роботу розглянуто й допущено до захисту на засіданні кафедри обчислювальної математики «29» травня 2023 р.,

протокол № 8

Завідувач кафедри

проф. Сергій Ляшко _____

Київ-2023

ЗМІСТ

ВСТУП

Спектральні та крайові задачі в математиці є важливими концепціями, які використовуються для вивчення властивостей диференціальних операторів та моделювання різних фізичних явищ.

Спектральні задачі зазвичай виникають при вивченні власних значень та власних функцій лінійних диференціальних операторів. Основними поняттями, пов'язаними зі спектральними задачами, є власні значення та власні функції. Власні значення - це значення параметра, при якому диференціальний оператор має нетривіальні розв'язки. Власні функції - це функції, які відповідають власним значенням та мають важливі фізичні або геометричні інтерпретації.

Крайові задачі виникають при розгляді поведінки розв'язків диференціальних рівнянь на межі заданої області. Вони включають у себе визначення крайових умов, що визначають поведінку розв'язків на межі області.

Спектральні задачі використовуються для вивчення власних частот та форм коливань різних фізичних систем, таких як струни, мембрани та резонатори. Окрім цього, власні значення та власні функції диференціальних операторів, таких як оператор Лапласа, використовуються для вивчення геометричних та топологічних властивостей просторів.

Крайові задачі використовуються для моделювання переносу тепла в матеріалах та визначення розподілу температури на границях системи.

Спектральні та крайові задачі відіграють важливу роль в різних галузях математики та науки. Знаходження власних значень та власних функцій оператора Лапласа є важливим завданням у математичній фізиці. Вони дозволяють вивчати коливання мембран, розподіл електромагнітного поля та інші явища у різних фізичних системах.

У геометрії та топології власні значення та власні функції оператора Лапласа пов'язані з геометричними та топологічними властивостями просторів. Вони дозволяють класифікувати форми, вивчати їх основні властивості та розв'язувати різні геометричні проблеми.

Моя дипломна робота ставить за мету розв’язання та дослідження власних значень оператора Лапласа на множині, що представляє собою квадрат, з якого прибрали круг (комірка з порожниною), тобто двовимірний випадок. На перших 3 власних значеннях оператора Лапласа буде досліджена залежність власних значень від радіусу прибраного круга. На границях області комірки з порожниною (як на границі квадрату так і на границі круга будуть задані умови Діріхле. Окрім цього будуть розглянуті кілька алгоритмів апроксимації задачі та порівняна їхня швидкодія.

РОЗДІЛ 1. ПОСТАНОВКА ЗАДАЧІ ЗНАХОДЖЕННЯ ВЛАСНИХ ЗНАЧЕНЬ ОПЕРАТОРА ЛАПЛАСА

1.1 ОСНОВНІ ВИЗНАЧЕННЯ

Перед тим як сформулювати задачу знаходження власних значень оператора Лапласа, важливо розглянути основні поняття та визначення.

Оператор Лапласа – це диференціальний оператор, який визначається як сума других часткових похідних другого порядку. Він широко використовується у математичній фізиці для моделювання різних фізичних явищ, таких як теплопровідність. Для двовимірного випадку записується таким чином:

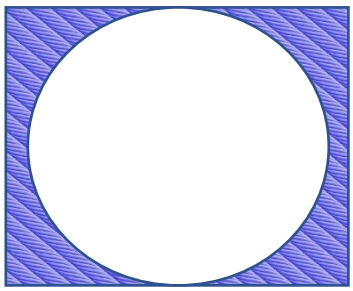
$$\Delta u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}$$

Крайова задача – це клас задач диференціальних рівнянь, які вимагають задання умов на границі області, в якій вони розглядаються. Крайова задача включає в себе постановку диференціального рівняння або системи рівнянь разом з крайовими умовами.

Умова Діріхле – це один із варіантів крайових умов, які використовуються для вирішення диференціальних рівнянь в області. У випадку оператора Лапласа та задач, пов'язаних з теплопровідністю, умова Діріхле вимагає фіксованого значення функції на границі області. В нашому випадку значення функції $u(x, y)$ на границі області дорівнює нулю. Записується це наступним чином:

$$u(x, y)|_\Gamma = 0$$

Комірка з порожниною – це область, яка має вигляд квадрату зі сторонами довжини *a*, з якого викинули круг радіуса не більше ніж *a/2*.



1.2 ПОСТАНОВКА ЗАДАЧІ

Постановка двовимірної задачі знаходження власних значень оператора Лапласа, відомого також як задача на власні значення Лапласа, полягає в пошуку параметрів, при яких оператор Лапласа має нетривіальні розв'язки.

Формальна постановка задачі знаходження власних значень оператора Лапласа у двовимірному випадку на області Γ з умовами Діріхле може бути записана наступним чином:

Знайти власні значення λ та власні функції $u(x,y)$ такі, що задовольняють рівнянню Лапласа та граничним умовам Діріхле на області Γ:

$$\Delta u(x,y) = -\lambda f(x,y),$$

$$u|_{\Gamma} = 0$$

$$\text{де } (x,y) \in \Gamma$$

$$\Gamma = [0,X] \times [0,Y]$$

де λ є параметром, який представляє власне значення оператора Лапласа, $u(x,y)$ є власною функцією, яка залежить від змінних x та y, Ω є областю у двовимірному просторі.

Метою задачі є знаходження значень параметра λ та відповідних власних функцій $u(x,y)$, які задовольняють рівнянню Лапласа і заданим граничним умовам Діріхле. Значення λ називають власними числами, а відповідні функції u(x, y) називають власними функціями оператора Лапласа.

РОЗДІЛ 2. АНАЛІТИЧНІ РОЗВ'ЯЗКИ ЗАДАЧІ ЗНАХОДЖЕННЯ ВЛАСНИХ ЗНАЧЕНЬ

2.1 МЕТОД РОЗДІЛЕННЯ ЗМІННИХ

Метод розділення змінних для знаходження власних значень оператора Лапласа у двовимірному випадку використовується для спрощення розв'язання цієї задачі. Згідно цьому методу, власна функція $u(x,y)$ представляється у вигляді суми:

$$u(x,y) = \nu_1(x,y) + \nu_2(x,y)$$

А задача

$$\Delta u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0$$

розбивається на 2 під задачі, а саме:

$$\frac{\partial^2 \nu_1}{\partial x^2} + \frac{\partial^2 \nu_1}{\partial y^2} = 0$$

$$\frac{\partial^2 \nu_2}{\partial x^2} + \frac{\partial^2 \nu_2}{\partial y^2} = 0$$

$$\nu_1 = X(x)Y(y)$$

Обидві задачі розв'язуються окремо та аналогічно. Розглянемо розв'язок на прикладі ν_1.

Підставляючи $\nu_1 = X(x)Y(y)$ у початкове рівняння і поділивши обидві частини на $(x)Y(y)$, отримаємо наступне:

$$\frac{X''(X)}{X(X)} = -\frac{Y''(x)}{Y(y)} = const = -\lambda$$

З цього випливає, що:

$$X''^{(x)} + \lambda X(X) = 0$$
$$Y''(y) - \lambda Y(y) = 0$$

З крайових умов Діріхле отримаємо, $X(0) = X(a) = 0$. В результаті отримано крайову задачу Штурма-Ліувілля:

$$X''(x) + \lambda X(X) = 0$$

$$X(0) = X(a) = 0$$

Якщо $\lambda < 0$, то загальний розв’язок рівняння $X''^{(x)} + \lambda X(X) = 0$ має вигляд:

$$X(x) = C_1 e^{\sqrt{-\lambda}x} + C_2 e^{-\sqrt{-\lambda}x}$$

З умови $X(0) = X(a) = 0$, отримаємо систему:

$$\begin{cases} C_1 + C_2 = 0 \\ C_1 e^{\alpha} + C_2 e^{-\alpha} = 0 \end{cases}$$

Оскільки $\alpha > 0$ та $\alpha \in \mathbb{R}$, наша система буде справною лише при $C_1 = 0$ та $C_2 = 0$. Тобто, $X(0) \equiv 0$, отже в даному випадку не існує нетривіальних розв'язків.

Якщо $\lambda = 0$, то загальний розв’язок рівняння $X''(x) + \lambda X(X) = 0$ має вигляд:

$$X(x) = C_1 x + C_2$$

З крайових умов отримаємо:

$$\begin{cases} c_2 = 0 \\ c_1 a = 0 \end{cases}$$

Тобто знову $X(0) \equiv 0$ і, як наслідок, тривіальний випадок.

Нарешті, при $\lambda > 0$ загальний розв’язок матиме вигляд:

$$X(x) = c_1 \, sin\left(\sqrt{\lambda}x\right) + c_2 \, cos\left(\sqrt{\lambda}x\right)$$

З крайових умов дістанемо:

$$\begin{cases} c_2 = 0 \\ C_1 \sin(\sqrt{\lambda}a) = 0 \end{cases}$$

Оскільки нас цікавлять нетривіальні випадки, то розглядатимемо випадок, коли $c_1 \neq 0$, тобто при:

$$\sin(\sqrt{\lambda}a) = 0$$

З цього випливає, що

$$\sqrt{\lambda} = \frac{\pi n}{a}$$

$$\lambda = \lambda_k = \left(\frac{\pi n}{a}\right)^2$$

Таким чином було отримано власні значення оператора Лапласа з умовами Діріхле.

РОЗДІЛ 3. НАБЛИЖЕНІ РОЗВ'ЯЗКИ ЗАДАЧІ ЗНАХОДЖЕННЯ ВЛАСНИХ ЗНАЧЕНЬ

3.1 Метод скінченних різниць

Метод скінченних різниць (finite difference method) це чисельний метод розв'язання диференціальних рівнянь. Він грунтується на наближенні похідних та інтегралів дискретними різницями. Метод широко використовується для чисельного моделювання різних фізичних процесів, включаючи розв'язання задач на власні значення.

Розглянемо задачу на власні значення оператора Лапласа з умовами Діріхле у двовимірному випадку. Маємо наступну постановку задачі:

Розглядаємо обмежену область D в двовимірному просторі, наприклад, квадрат або коло.

Потрібно знайти функцію $u(x, y)$ яка задовольняє рівняння Лапласа: $\Delta u(x, y) = 0$всередині області D.

Задані граничні умови: $\mathrm{u(x, y) = f(x, y)}$ на границі області D, де $f(x, y)$ – деяка задана функція. Щоб застосувати метод скінченних різниць до цієї задачі, необхідно розбити область D на сітку з обмеженими вузлами. Нехай маємо $M + 1$ вузлів по осям x і y. Кроки сітки будуть $\mathrm{h_x = L_x/M}$ і $\mathrm{h_y = L_y/M}$, де $\mathrm{L_x}$ і $\mathrm{L_y}$ – довжини сторін області D вздовж вісей x і y відповідно.

Введемо вузли сітки, відповідні точкам $\mathrm{(x_i, y_j)}$, де i = 0, 1, ..., M і j = 0, 1, ..., M. Функція $u(x, y)$ апроксимується значеннями на цих вузлах: $\mathrm{(x_i, y_j) = u_{i,j}}$.

Для апроксимації другої похідної оператора Лапласа використовуються центральні різниці. За допомогою цих різниць можна записати рівняння Лапласа на сітці. Для точки (x_i, y_j) отримаємо:

$$\Delta u(x_i, y_j) \approx (u_{i+1,j} - 2u_{i,j} + u_{i-1,j})/h_x^2 + (u_{i,j+1} - 2u_{i,j} + u_{i,j-1})/h_y^2 = 0$$

Дане рівняння буде виконуватися для всіх внутрішніх вузлів сітки. Застосувавши це апроксимаційне рівняння до кожного внутрішнього вузла сітки, отримуємо систему лінійних рівнянь для невідомих значень $u_{i,j}$.

Граничні умови Діріхле можуть бути враховані шляхом присвоєння відповідних значень на границі сітки. Наприклад, якщо на границі вузлам (x_i, y_0) та (x_i, y_M)відповідають граничні умови, то відповідні рівняння в системі будуть мати вигляд:

$$u_{i,0} = f\ (x_i, y_0),$$

$$u_{i,M} = f\ (x_i, y_M).$$

Отриману систему лінійних рівнянь можна розв'язати чисельними методами, такими як метод Гаусса або ітераційні методи (наприклад, метод простої ітерації або метод Зейделя). Розглянемо метод Гаусса:

Перетворюємо систему лінійних рівнянь на розширену матрицю і вирішуємо її за допомогою методу Гаусса для отримання невідомих значень власних функцій на внутрішніх вузлах сітки. В результаті отримаємо власні значення і власні функції. Використовуючи отримані невідомі значення власних функцій на внутрішніх вузлах сітки, обчислюємо власні функції на всій області D. Також обчислюємо власні значення, які відповідають цим власним функціям. Чим більше вузлів у сітці, тим більша точність отриманого розв'язку.

3.2 МЕТОД ОРТОГОНАЛЬНИХ ПРОЕКЦІЙ

Цей метод базується на розкладі функції $u(x, y)$ у базисі ортогональних функцій. У даному випадку ми використовуємо ортогональні функції, що відповідають власним значенням оператора Лапласа з умовами Діріхле. Оскільки ми розглядаємо двовимірний випадок, такими функціями є функції синусів і косинусів.

Нехай $\varphi_n(x, y)$ буде ортогональною функцією, що відповідає n-му власному значенню λ_n оператора Лапласа з умовами Діріхле. Тоді розклад функції u(x,y) у цьому базисі матиме вигляд:

$$\mathrm{u(x,y)} = \mathrm{c}_1\varphi_1(\mathrm{x,y}) + \mathrm{c}_2\varphi_2(x, y) \ldots + c_{\mathrm{n}}\varphi_n(x, y)$$

де c_1, c_2, .. , c_{n} – коефіцієнти розкладу.

Застосуємо метод ортогональних проекцій, щоб знайти ці коефіцієнти. Помножимо обидві частини рівняння Лапласа на функцію $\varphi_m(x, y)$ і проінтегруємо по всьому двовимірному простору. Використовуючи ортогональність функцій $\varphi_n(x, y)$ та $\varphi_m(x, y)$ всі доданки зникнуть, за винятком одного, де n=m. Отримаємо таке рівняння для коефіцієнтів c_m:

$$\lambda_{\mathrm{m}}\mathrm{c_m} = \int\int (\Delta \mathrm{u})\, \varphi_m(x, y)\mathrm{dxdy}$$

Для того, щоб знайти коефіцієнти c_m , потрібно обчислити значення подвійного інтеграла і також знайти власні значення. Це можна зробити як аналітично, так і наближено. Після обчислення коефіцієнтів, ми можемо побудувати розв'язок задачі на власні значення, використовуючи розклад функції u(x,y) у базисі ортогональних функцій.

3.3 ВАРІАЦІЙНИЙ МЕТОД

Варіаційний метод є потужним інструментом для розв'язання різноманітних математичних задач, зокрема задач на власні значення лінійних операторів. Він полягає у знаходженні мінімуму чи максимуму наступного функціоналу:

$$J(u) = \frac{1}{2}\int_{\Omega} |\nabla u|^2 \, dx \, dy - \int_{\Omega} \lambda u^2 \, dx \, dy$$

де Ω – це область, на якій задана двовимірна задача на власні значення оператора Лапласа, а ∇u – це градієнт, тобто вектор вигляду $\left(\frac{\partial u}{\partial x}, \frac{\partial u}{\partial y}\right)$. Суть в тому, щоб знайти функції u та числа λ, при яких оператор $J(u)$ досягає мінімуму чи максимуму (в нашому випадку мінімуму).

Для застосування варіаційного методу, ми вводимо варіацію функції u позначену як δu і розглядаємо наступне рівняння:

$$\delta J(u, \delta u) = 0$$

δJ – це варіація функціоналу J. Ми шукаємо таку варіацію δu при якій написане вище рівняння має розв'язок для довільної функції u.

Розглянемо перший доданок в правій частині, а саме $\frac{1}{2}\int_{\Omega}|\nabla u|^2 \, dx \, dy$. Якщо скористатись формулою Гріна, то можна цей доданок записати у вигляді:

$$-\frac{1}{2}\int_{\Omega} u\Delta u \, dx \, dy \; + \frac{1}{2}\int_{\partial\Omega} \frac{\partial u}{\partial n} u \, ds$$

Тут $\frac{\partial u}{\partial n}$-- зовнішня нормаль. $\partial\Omega$ – границя області.

Підставивши це в початкове рівняння отримаємо:

$$-\frac{1}{2}\int_{\Omega} u\Delta u\,dx\,dy\; +\frac{1}{2}\int_{\partial\Omega}\frac{\partial u}{\partial n}u\,ds\;\; -\int_{\Omega}\lambda u^2\,dx\,dy\;=\;0$$

Застосувавши принцип варіаційного методу, отримаємо:

$$\frac{1}{2}\int_{\Omega}\delta u\Delta u\,dx\,dy\; -\frac{1}{2}\int_{\partial\Omega}\frac{\partial u}{\partial n}\delta u\,ds\;\; -\int_{\Omega}\lambda\delta u\,dx\,dy\;=\;0$$

Оскільки це рівняння має виконуватись для довільної варіації δu, застосуємо формулу Гріна знову до першого доданку і, врахувавши умови Діріхле, дістанемо:

$$\int_{\Omega}\nabla u\nabla\delta u\,dx\,dy\; -\int_{\partial\Omega}\frac{\partial u}{\partial n}\delta u\,ds\;\; -\int_{\Omega}\lambda\delta u\,dx\,dy\;=\;0$$

Щоб отримати розв'язок ми маємо обрати варіацію δu таким чином, щоб вона дорівнювала нулю на границі $\partial\Omega$. В такому випадку рівняння перетворюється на наступне:

$$\int_{\Omega}\nabla u\nabla\delta u\,dx\,dy-\int_{\Omega}\lambda\delta u\,dx\,dy\;=\;0$$

Отримана рівність є варіаційною формою задачі на власні значення оператора Лапласа з крайовими умовами Діріхле. Розв'язком цієї рівності будуть такі функції $u(x,y)$ та такі власні числа λ, при яких ця рівність буде виконуватись для довільної варіації δu.

Отже, застосовуючи варіаційний метод до задачі на власні значення оператора Лапласа з умовами Діріхле у двовимірному випадку, ми отримуємо систему варіаційних рівнянь, яку можна розв'язати аналітично або наближено і тим самим знайти власні числа λ та відповідні їм власні функції u(x,y).

РОЗДІЛ 4. ПРОГРАМНА РЕАЛІЗАЦІЯ НАБЛИЖЕНИХ АЛГОРИТМІВ РОЗВ'ЯЗКУ ЗАДАЧІ ЗНАХОДЖЕННЯ ВЛАСНИХ ЗНАЧЕНЬ

4.1 Реалізація методу скінченних різниць

Для реалізації було обрано мову програмування python. Алгоритм полягає у наступному:

Спочатку визначається функція solve_eigenvalues, яка обчислює власні значення матриці Лапласа для заданого розміру сітки n та радіуса вирізаного кола radius. Усередині функції виконуються наступні кроки:

- Створюється сітка X і Y з рівномірно розподіленими значеннями від 0 до 1.
- Створюється маска circle_mask, яка визначає точки, що знаходяться всередині або на межі кола з центром у (0.5, 0.5) та заданим радіусом. Це робиться шляхом обчислення відстані кожної точки до центру і порівняння її з радіусом.
- Виключаються точки, що знаходяться всередині кола, з сітки X і Y.
- Створюється відображення між точками сітки і індексами в матриці.
- Створюється матриця Лапласа L за допомогою функції diags з бібліотеки scipy.sparse. Матриця Лапласа використовується для моделювання дифузії на сітці і має розмірність, що відповідає кількості точок всередині кола.
- Обчислюються власні значення і вектори матриці L за допомогою функції eigsh з бібліотеки scipy.sparse.linalg. У цьому випадку обираються три найменших власних значення (num_eigenvalues = 3) за допомогою параметра which='SM', що означає "найменші".
- Функція повертає обчислені власні значення.

Далі встановлюються параметри:

- grid_size - розмір сітки.
- min_radius - мінімальний радіус вирізаного кола.
- max_radius - максимальний радіус вирізаного кола.
- step - крок зміни радіуса.

Після цього створюються порожні списки radii, lambda_1_values, lambda_2_values і lambda_3_values для зберігання радіусів та власних значень.

Ітерується по значеннях радіуса з radii і обчислюються власні значення для кожного радіуса:

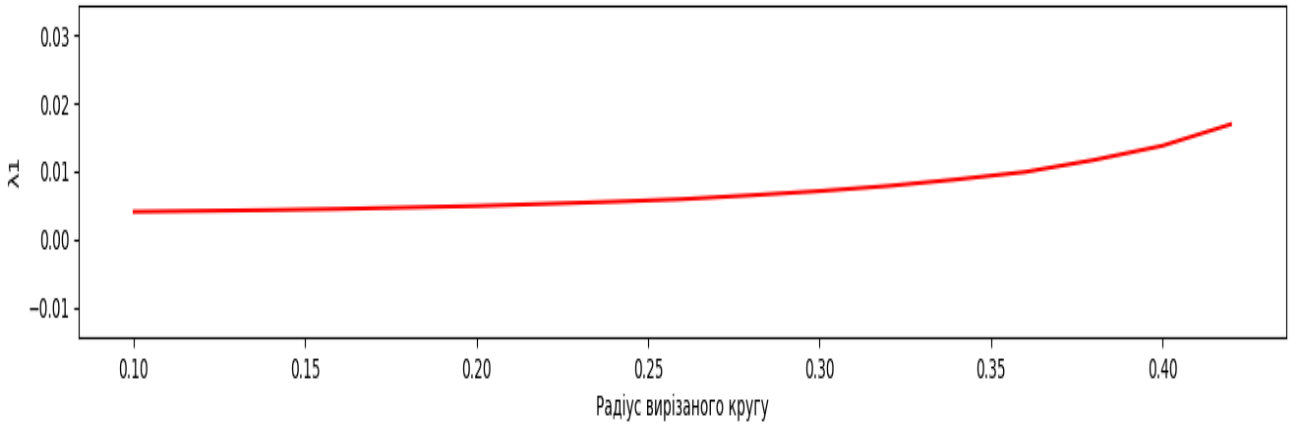
- Викликається функція solve_eigenvalues для обчислення власних значень для поточного радіуса.
- Обчислені власні значення додаються до відповідних списків lambda_1_values, lambda_2_values і lambda_3_values.
- Виводяться значення радіуса та власних значень в консоль.

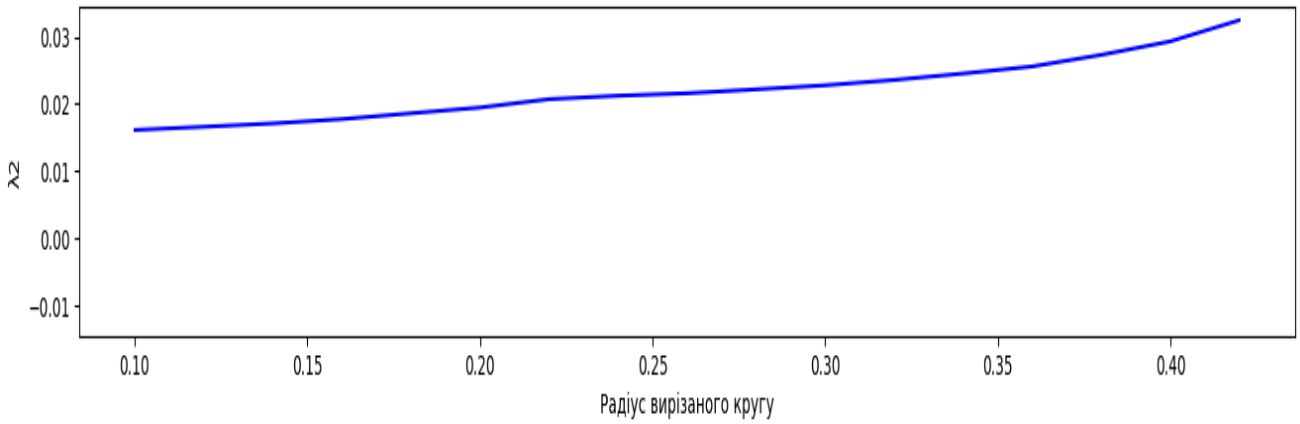
В кінці програми будуються графіки:

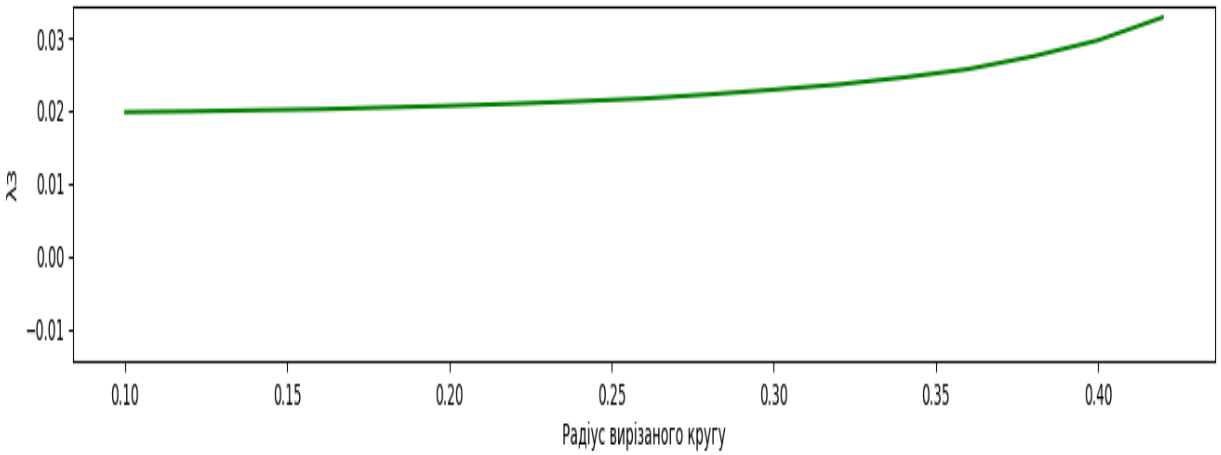
- Створюється фігура за допомогою plt.figure, встановлюється розмір 10x8 см.
- Створюються три підграфіки у вигляді вертикального розміщення за допомогою plt.subplot.
- Для кожного підграфіка будується графік залежності обчислених власних значень від радіуса за допомогою plt.plot.
- Встановлюються підписи осей та рівні масштаби осей за допомогою plt.xlabel, plt.ylabel та plt.axis('equal').
- Викликається plt.tight_layout, щоб забезпечити компактне розташування графіків.
- Виводиться графічне вікно за допомогою plt.show.

Результати роботи програми:

$radius$ / λ	0.1	0.16	0.2	0.26	0.3	0.36	0.4	0.42
λ_1	0.0040	0.0044	0.0049	0.0059	0.0071	0.0099	0.0137	0.0169
λ_2	0.0161	0.0178	0.0195	0.0216	0.0228	0.0256	0.0293	0.0325
λ_3	0.0198	0.0202	0.0207	0.0217	0.0229	0.0257	0.0297	0.0328







Алгоритмічна складність цього коду залежить від найскладнішої його частини. У цьому випадку найскладнішою частиною є обчислення власних значень та векторів для матриці Лапласа. Алгоритмічна складність цієї операції залежить від розміру матриці та кількості необхідних власних значень. Припустимо, що розмір матриці Лапласа становить n x n, а кількість необхідних власних значень дорівнює k. У такому випадку, алгоритмічна складність обчислення власних значень та векторів становить приблизно $O(n^2 \cdot k)$.

У даному прикладі кількість необхідних власних значень дорівнює 3. Таким чином, алгоритмічна складність даного коду становить приблизно $O(n^2 \cdot 3)$.

4.2 Реалізація методу ортогональних проекцій

Алгоритм методу ортогональних проекцій наступний:

Спочатку створюються точки всередині квадрата за допомогою функції linspace з бібліотеки numpy. Змінні x і y містять координати точок по вісі X і Y відповідно. Функція meshgrid використовує ці точки для створення двовимірної сітки X і Y, яка містить всі комбінації координат. Далі створюється матриця граничних умов boundary, яка має розмірність N (кількість точок сітки x). Ця матриця містить нулі, за винятком першого і останнього елементів, які встановлюються на значення 1. Вона

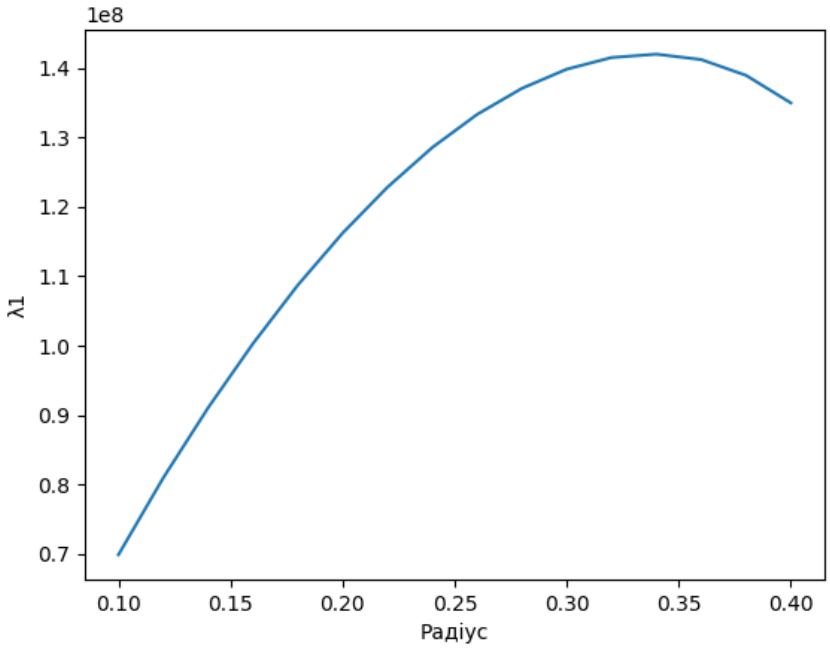
використовується для задання граничних умов в розв'язку. Після цього задаються порожні списки lambda_1_values, lambda_2_values і lambda_3_values для зберігання власних значень. Далі виконується ітерація по значеннях радіуса від 0.1 до 0.42 з кроком 0.02 за допомогою функції arange з бібліотеки numpy. У кожній ітерації виконуються наступні кроки:

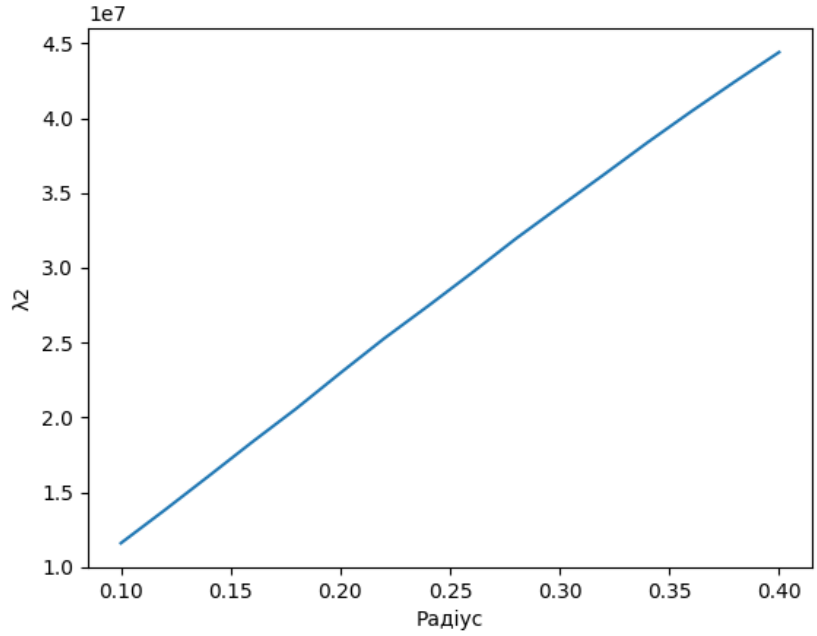
- Обчислюється рівняння кола за допомогою операторів ** (піднесення до степеня) і +.
- Створюється маска mask, яка відділяє область поза колом шляхом порівняння рівняння кола з квадратом радіусу.
- Створюється матриця Лапласа L для внутрішньої області шляхом встановлення значень для кожного внутрішнього вузла. Застосовується формула для чисельного обчислення другої похідної.
- Застосовується маска до матриці Лапласа, встановлюючи значення, що відповідають точкам поза колом, на велике значення (у цьому випадку 1e6).
- Застосовуються граничні умови до матриці Лапласа шляхом заміни відповідних значень нулями.
- Встановлюються спеціальні умови для точки в центрі матриці, де N//2 відповідає індексу середньої точки.
- Обчислюються власні значення матриці Лапласа за допомогою функції eigvals з бібліотеки scipy.linalg.
- Власні значення сортуються за зростанням.
- Відкидається уявна частина власних значень за допомогою функції real з бібліотеки numpy.
- Власні значення округлюються до двох знаків після коми за допомогою функції round з бібліотеки numpy.
- Отримані власні значення додаються до відповідних списків lambda_1_values, lambda_2_values і lambda_3_values.

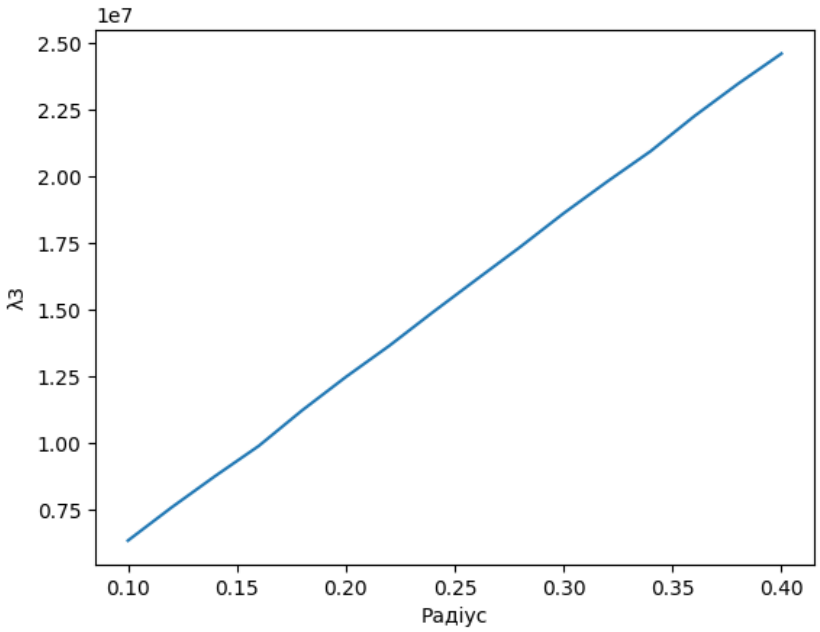
Після закінчення ітерації будуються графіки:

- Створюється масив radii, який містить значення радіусів з ітерації.
- Для кожного власного значення будується графік залежності від радіуса за допомогою функції plot з бібліотеки matplotlib.pyplot.
- Встановлюються підписи вісей і назви графіків за допомогою функцій xlabel, ylabel і title.
- Виводяться графіки за допомогою функції show з бібліотеки matplotlib.pyplot.
- Результати роботи програми:

$radius$ / λ	0.1	0.16	0.2	0.26	0.3	0.36	0.4
λ_1	69896408.58	100293176.02	116211563.66	133300501.72	139818600.74	141204650.58	134964747.92
λ_2	11596853.31	18370409.88	22963556.88	29663266.72	34087247.08	40451560.87	44408665.72
λ_3	6357321.71	9899695.18	12485004.91	16139191.75	18618605.25	22258517.51	24601126.37


1e8
1.4
1.3
1.2
1.1
1.0
0.9
0.8
0.7
λ1
0.10
0.15
0.20
0.25
0.30
0.35
0.40
Радіус


1e7
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
λ2
0.10
0.15
0.20
0.25
0.30
0.35
0.40
Радіус



Складність алгоритму: загальна алгоритмічна складність цього коду становить приблизно O(N^3).

4.3 Реалізація варіаційного методу

Спочатку визначаємо параметри та геометрію простору:

- n - кількість точок на стороні квадрата.
- h - крок сітки, обернений до n.
- radius_range - змінний діапазон радіусів, який складається з значень, округлених до двох знаків після коми, у межах від 0.1 до 0.42 з кроком 0.02.

Далі йде створення списків для зберігання радіусів та власних значень. Після цього ітерація по значеннях радіуса в radius_range:

- Створення сітки всередині квадрата зі зміненим радіусом. Використовуються функції np.meshgrid та математичне вираз (x - 0.5)**2 + (y - 0.5)**2 >= radius**2, щоб отримати індекси точок, які знаходяться поза колом радіусом `radius` із центром у (0.5, 0.5).

- Після цього йде створення матриці Лапласа A розміром (n**2, n**2). Перебираються всі точки на сітці, і для кожної точки перевіряється, чи знаходиться вона всередині кола. Якщо так, то відповідний елемент матриці A заповнюється значеннями згідно формули для матриці Лапласа у двовимірному випадку.
- Далі вирішення задачі знаходження власних значень для матриці A за допомогою функції eigh з обмеженням лише на перші три власні значення.
- Додавання поточного радіуса та відповідних власних значень до списків radii, eigenvalues1, eigenvalues2 і eigenvalues3.

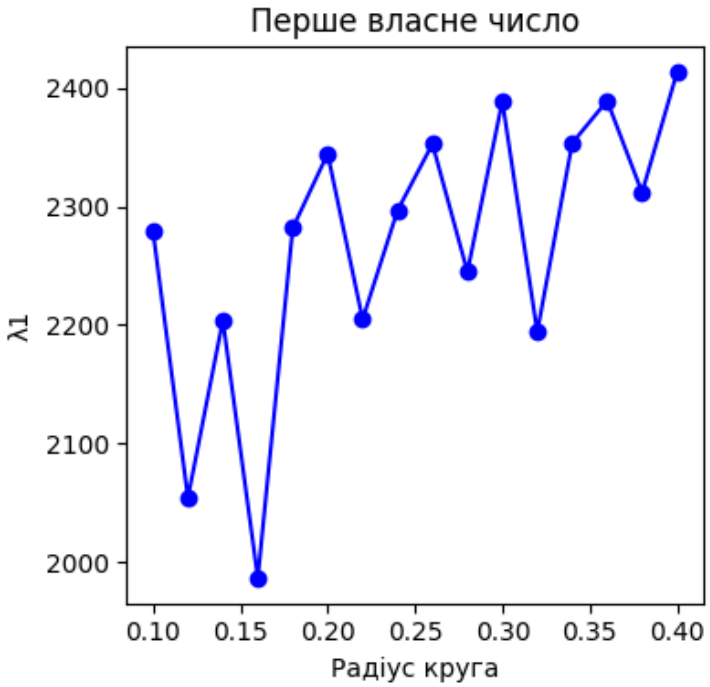
Наприкінці виведення значень трьох власних чисел для кожного радіуса та побудова графіків за допомогою matplotlib.pyplot

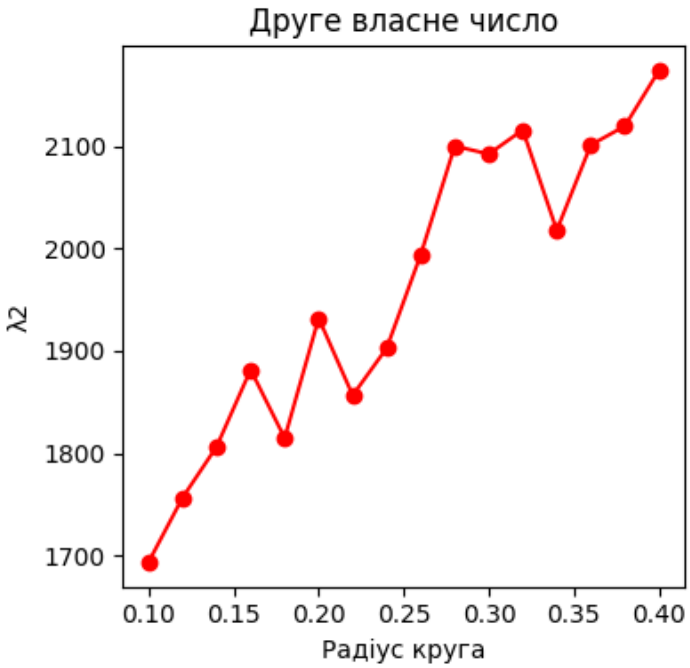
- Створення фігури з трьома підграфіками.
- Побудова графіка першого власного числа λ1 залежно від радіуса.
- Побудова графіка першого власного числа λ2 залежно від радіуса.
- Побудова графіка першого власного числа λ3 залежно від радіуса.
- Налаштування міток осей та заголовків графіків.
- Відображення графіків.

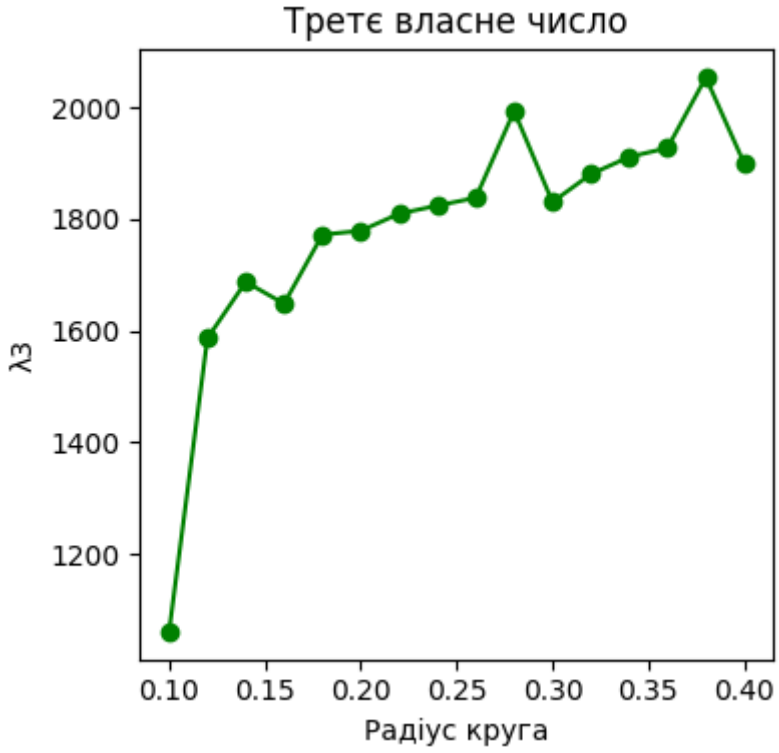
Округлення радіусів та власних значень до двох знаків після коми виконується за допомогою функції round. Програма виконує цикл для різних радіусів та будує графіки залежності власних чисел від радіуса для перших трьох власних чисел.

Результати роботи програми:

$radius$ / λ	0.1	0.16	0.2	0.26	0.3	0.36	0.4
λ_1	2279.22	1986.30	2344.69	2353.02	2389.08	2389.50	2413.53
λ_2	1693.78	1881.67	1931.97	1993.91	2092.21	2101.12	2173.57
λ_3	1060.64	1648.16	1778.83	1837.87	1830.26	1927.23	1899.57







Складність цього алгоритму становить приблизно O($k \cdot N^3$), де K - кількість значень в змінному діапазоні радіусів, а N - кількість точок на стороні квадрата.

ВИСНОВКИ

У цій роботі було розглянуто спектральні еліптичні та крайові задачі на комірках з порожниною. Були розібрані як аналітичні так і наближені методи розв'язання задач даного типу. Для наближеного розв'язку задачі на власні числа оператора Лапласа з початковими умовами Діріхле було реалізовано на мові програмування python наступні алгоритми: метод скінченних різниць, метод ортогональних проекцій, варіаційний метод. Також мало місце порівняння складності роботи усіх трьох вищезгаданих алгоритмів. У контексті двовимірної задачі знаходження власних значень оператора Лапласа з умовами Діріхле в Python, метод скінченних різниць виявився найшвидшим. Метод ортогональних проекцій та варіаційний метод також можуть застосовуватися для розв'язання даної задачі, але вони повільніші, та, як правило використовуються для більш загальних задач, таких як задач оптимізації та мінімізації функціоналів.

ЛІТЕРАТУРА

[1] Walter A. Strauss "Partial Differential Equations: An Introduction". B: *John Wiley & Sons, 2007*. ISBN: 978-0470-05456-7, c. 422-427

[2] Qing Han and Fanghua Lin "Elliptic Partial Differential Equations". B: *Courant Institute of Mathematical Sciences, New York University, 2011.* ISBN: 978-0-8218-5313-9, c. 1-19

[3] Lawrence C. Evans "Partial Differential Equations". B: *American Mathematical Society, 2010.* ISBN: 978-082-18-2973-2, c. 20-33

[4] Stanley J. Farlow "Partial Differential Equations for Scientists and Engineers". B: *Dover Publications, 2012.* ISBN: 978-048-66-7620-3, c. 299-317

[5] Mark S. Gockenbach "Partial Differential Equations: Analytical and Numerical Methods" B: *Society for Industrial and Applied Mathematics (SIAM), 2011*. ISBN: 978-048-66-7620-3, c. 593-614

[6] Shuenn-Yih Chang "Partial Differential Equations: Methods, Applications and Theories". B: *WSPC, 2019*. ISBN 978-9811202230, c. 171-182

[7] John C. Strikwerda "Finite Difference Schemes and Partial Differential Equations" . B: *Society for Industrial and Applied Mathematics (SIAM), 2004.* ISBN: 978-089-87-1567-5, c. 61-94

[8] G. D. Smith "Numerical Solution of Partial Differential Equations: Finite Difference Methods". B: *Oxford University Press, 1986.* ISBN 978-019-85-9650-9, c. 210-217

[9] Randall J. LeVeque "Finite Difference Methods for Ordinary and Partial Differential Equations: Steady-State and Time-Dependent Problems". B: *Society for Industrial and Applied Mathematics (SIAM), 2007.* ISBN: 978-089-87-1629-0, c. 150-173

[10] Ferziger, J.H. and Perić, M. "Finite Difference Methods for Computational Fluid Dynamics: A Practical Guide". В: *Springer, 2012.* ISBN: 978-364-21-6053-0, с. 224-241

[11] Michael Struwe "Variational Methods: Applications to Nonlinear Partial Differential Equations and Hamiltonian Systems" В: *Springer, 2008.* ISBN: 978-354-07-4012-4, с. 96-105

[12] С.О. Войцеховський, В.І. Гаркуша, М.П. Копистира та ін. «Методичні розробки до вивчення нормативного курсу «Рівняння математичної фізики» (класифікація рівнянь другого порядку, формули Даламбера і Пуассона, метод відокремлення змінних) для студентів факультету кібернетики», Київ 2001 р, с. 26-38

ДОДАТОК А. Код програм.

Метод скінченних різниць:

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.sparse import diags
from scipy.sparse.linalg import eigsh
def solve_eigenvalues(n, radius):
    # Створення сітки
    x = np.linspace(0, 1, n)
    y = np.linspace(0, 1, n)
    X, Y = np.meshgrid(x, y)
    # Створення маски для вирізаного круга
    circle_mask = np.sqrt((X - 0.5)**2 + (Y - 0.5)**2) <= radius
    # Виключення точок всередині круга з сітки
    X = X[~circle_mask]
    Y = Y[~circle_mask]
    # Створення відображення з сітки в матрицю
    num_points = X.size
    point_to_index = np.arange(num_points)
    index_to_point = np.zeros(n**2, dtype=int)
    index_to_point[~circle_mask.flatten()] = np.arange(num_points)
```

```
    # Створення матриці Лапласа
    # Створення матриці Лапласа
    L = diags([-1, -1, 4, -1, -1], [-n, -1, 0, 1, n], shape=(num_points, num_points))
    L = L.toarray()
    # Знаходження власних значень та векторів
    num_eigenvalues = 3
    eigenvalues, eigenvectors = eigsh(L, k=num_eigenvalues, which='SM')

    return eigenvalues

    return eigenvalues

# Задаємо параметри
grid_size = 50  # Розмір сітки
min_radius = 0.1  # Мінімальний радіус
max_radius = 0.4  # Максимальний радіус
step = 0.02  # Крок зміни радіуса

# Зберігання радіусів та власних значень
radii = np.arange(min_radius, max_radius + step, step)
lambda_1_values = []
lambda_2_values = []
```

```
lambda_3_values = []


# Ітеруємося по значенням радіуса та розв'язуємо задачу для кожного значення
for radius in radii:
    eigenvalues = solve_eigenvalues(grid_size, radius)
    lambda_1_values.append(eigenvalues[0])
    lambda_2_values.append(eigenvalues[1])
    lambda_3_values.append(eigenvalues[2])


    # Вивід власних значень та радіуса в консоль
    print(f"Радіус: {radius:.2f}")
    print(f"λ1: {eigenvalues[0]:.4f}")
    print(f"λ2: {eigenvalues[1]:.4f}")
    print(f"λ3: {eigenvalues[2]:.4f}")
    print()


# Побудова графіків
plt.figure(figsize=(10, 8))


plt.subplot(3, 1, 1)
plt.plot(radii, lambda_1_values, color='red')
plt.xlabel('Радіус вирізаного кругу')
```

```
plt.ylabel('λ1')

plt.axis('equal')

plt.ylim(0, 0.02)


plt.subplot(3, 1, 2)

plt.plot(radii, lambda_2_values, color='blue')

plt.xlabel('Радіус вирізаного кругу')

plt.ylabel('λ2')

plt.axis('equal')

plt.ylim(0, 0.02)


plt.subplot(3, 1, 3)

plt.plot(radii, lambda_3_values, color='green')

plt.xlabel('Радіус вирізаного кругу')

plt.ylabel('λ3')

plt.axis('equal')

plt.ylim(0, 0.02)


plt.tight_layout()

plt.show()
```

Метод ортогональних проекцій:

```
import numpy as np
```

```
import matplotlib.pyplot as plt
from scipy import linalg

# Створення точок всередині квадрата
x = np.linspace(0, 1, 500)
y = np.linspace(0, 1, 500)
X, Y = np.meshgrid(x, y)

# Створення матриці граничних умов
N = len(x)
boundary = np.zeros(N)
boundary[0] = 1
boundary[-1] = 1

# Списки для зберігання власних значень
lambda_1_values = []
lambda_2_values = []
lambda_3_values = []

# Ітерація по значенням радіуса
for radius in np.arange(0.1, 0.42, 0.02):
    # Рівняння кола
```

```
circle = (X - 0.5) ** 2 + (Y - 0.5) ** 2


# Створення маски для виділення області поза колом
mask = np.where(circle <= radius**2, True, False)


# Створення матриці Лапласа для внутрішньої області
L = np.zeros_like(X)
L[1:-1, 1:-1] = -4
L[1:-1, :-2] += 1
L[1:-1, 2:] += 1
L[:-2, 1:-1] += 1
L[2:, 1:-1] += 1
L *= 1 / (x[1] - x[0])**2


# Застосування маски до матриці Лапласа
L = np.where(mask, L, 1e6)


# Застосування граничних умов до матриці Лапласа
L[0, :] = 0
L[-1, :] = 0
L[:, 0] = 0
L[:, -1] = 0
```

```
L[N//2, N//2] = 1

L[N//2, N//2-1] = -1


# Розв'язок власних значень

eigenvalues = linalg.eigvals(L)


# Сортування власних значень за зростанням

eigenvalues = np.sort(eigenvalues)


# Зміна від'ємних власних чисел на додатні

eigenvalues = np.abs(eigenvalues)


# Округлення власних значень до 2 знаків після коми

eigenvalues = np.round(eigenvalues, 2)


# Додавання власних значень до відповідних списків

lambda_1_values.append(eigenvalues[0])

lambda_2_values.append(eigenvalues[1])

lambda_3_values.append(eigenvalues[2])


# Округлення радіуса до 2 знаків після коми

radius = round(radius, 2)
```

```
    # Вивід власних значень для поточного радіуса
    print(f"Радіус: {radius}")
    print(f"лямбда_1 = {eigenvalues[0]}")
    print(f"лямбда_2 = {eigenvalues[1]}")
    print(f"лямбда_3 = {eigenvalues[2]}")
    print()

# Побудова графіків
radii = np.arange(0.1, 0.42, 0.02)

# Графік lambda_1
plt.figure()
plt.plot(radii, lambda_1_values)
plt.xlabel('Радіус')
plt.ylabel('λ1')

# Графік lambda_2
plt.figure()
plt.plot(radii, lambda_2_values)
plt.xlabel('Радіус')
plt.ylabel('λ2')
```

```
# Графік lambda_3
plt.figure()
plt.plot(radii, lambda_3_values)
plt.xlabel('Радіус')
plt.ylabel('λ3')

plt.show()
```

Варіаційний метод:

```
import numpy as np
from scipy.linalg import eigh
import matplotlib.pyplot as plt

# Визначення параметрів та геометрії
n = 50  # Кількість точок на стороні квадрата
h = 1 / n  # Крок сітки
radius_range = np.round(np.arange(0.1, 0.42, 0.02), 2)  # Змінний діапазон радіусів

# Створення списків для радіусів та власних значень
radii = []
eigenvalues1 = []
eigenvalues2 = []
```

```
eigenvalues3 = []


for radius in radius_range:
    # Створення сітки всередині квадрата зі зміненим радіусом
    x, y = np.meshgrid(np.linspace(0, 1, n), np.linspace(0, 1, n))
    circle_indices = (x - 0.5)**2 + (y - 0.5)**2 >= radius**2


    # Створення матриці Лапласа
    A = np.zeros((n**2, n**2))
    for i in range(n):
        for j in range(n):
            if not circle_indices[i, j]:
                index = i * n + j
                A[index, index] = 4 / h**2
                if i > 0:
                    A[index, index - n] = -1 / h**2
                if i < n - 1:
                    A[index, index + n] = -1 / h**2
                if j > 0:
                    A[index, index - 1] = -1 / h**2
                if j < n - 1:
                    A[index, index + 1] = -1 / h**2
```

```
    # Вирішення задачі знаходження власних значень
    eigenvalues, _ = eigh(A, subset_by_index=(0, 2))  # Вибір лише перших трьох власних значень

    # Додавання радіуса та власних значень до відповідних списків
    radii.append(radius)
    eigenvalues1.append(abs(eigenvalues[0]))
    eigenvalues2.append(abs(eigenvalues[1]))
    eigenvalues3.append(abs(eigenvalues[2]))

    # Виведення значень трьох власних чисел для поточного радіусу
    print(f"Радіус: {radius:.2f}")
    print(f"λ1 = {np.abs(eigenvalues[0]):.4f}")
    print(f"λ2 = {np.abs(eigenvalues[1]):.4f}")
    print(f"λ3 = {np.abs(eigenvalues[2]):.4f}")
    print()

# Побудова графіків
plt.figure(figsize=(12, 4))

# Графік першого власного числа (λ1)
plt.subplot(131)
```

```
plt.plot(radii, eigenvalues1, 'bo-')
plt.xlabel('Радіус круга')
plt.ylabel('λ1')
plt.title('Перше власне число')

# Графік другого власного числа (λ2)
plt.subplot(132)
plt.plot(radii, eigenvalues2, 'ro-')
plt.xlabel('Радіус круга')
plt.ylabel('λ2')
plt.title('Друге власне число')

# Графік третього власного числа (λ3)
plt.subplot(133)
plt.plot(radii, eigenvalues3, 'go-')
plt.xlabel('Радіус круга')
plt.ylabel('λ3')
plt.title('Третє власне число')

plt.tight_layout()
plt.show()
```

Відгук наукового керівника
на кваліфікаційну роботу бакалавра на тему:
«Спектральні та крайові еліптичні задачі на комірках з порожнинами»
студента 4-го курсу бакалаврату факультету
комп'ютерних наук та кібернетики
Київського національного університету імені Тараса Шевченка
Ларіонова Андрія Віталійовича

Рецензована робота присвячена дослідженню спектральних та крайових еліптичних задач на комірках з порожнинами.

Останнім часом задачі на комірках з порожнинами представляють значний інтерес, оскільки вони дозволяють моделювати та досліджувати динамічні процеси фільтрації рідин у пористих середовищах, що є важливим при плануванні використання підземних ресурсів, експлуатації гребель і гідроелектростанцій. Дослідження таких процесів методами інженерних спостережень на практиці є складним і дорогим. Таким чином, моделювання є дійсним способом прогнозування і оптимізації методів раціональної експлуатації гребель і гідроелектростанцій, водо, газо, нафтовидобування, використання, очищення і запобігання забрудненню підземних вод та ресурсів.

У роботі студент розглянув три алгоритми знаходження наближеного розв'язку спектральної задачі оператора Лапласа з крайовими умовами Діріхле на комірці з порожниною у двовимірному випадку. Був наведений детальний опис програмної реалізації кожного з алгоритмів на мові програмування python. Результатом роботи є графіки та таблиці залежності власних значень від радіусу порожнини, таким чином показана залежність власних значень від області, на якій розглядається задача. Окрім цього, була дана оцінка алгоритмічної складності кожного з алгоритмів та був встановлений оптимальний алгоритм для даної задачі.

Вважаю, що кваліфікаційна робота студента Андрія Ларіонова відповідає вимогам, які висуваються до бакалаврських робіт, і заслуговує на оцінку «добре» (80), а її автор заслуговує на присвоєння кваліфікації бакалавра.

Професор кафедри обчислювальної математики
факультету комп'ютерних наук та кібернетики
Київського національного університету
імені Тараса Шевченка,
доктор фізико-математичних наук, професор  Володимир СЕМЕНОВ

Рецензія на кваліфікаційну роботу бакалавра на тему:
«Спектральні та крайові еліптичні задачі на комірках з порожнинами»
студента 4-го курсу бакалаврату факультету
комп'ютерних наук та кібернетики
Київського національного університету імені Тараса Шевченка
Ларіонова Андрія Віталійовича

Задачі на знаходження власних значень є дуже важливим та потужним інструментом як в теоретичній так і прикладній математиці.

Метою кваліфікаційної роботи Андрія Ларіонова є дослідження задачі на власні значення оператора Лапласа з крайовими умовами Діріхле у двовимірному випадку на області, яка представляє собою квадрат, з внутрішньої частини якого прибрали круг. Для розв'язування цієї задачі було використано як варіаційний та і сітковий підхід. Це дуже вдалий вибір, оскільки він показує, що студент підійшов до задачі з різних сторін.

За змістом роботи можна зробити зауваження. Бажано навести більш розгорнутий теоретичний опис поняття комірок з порожнинами та задач, які розглядаються на них. У роботі не розглянуто теоретичні основи оцінок алгоритмічної складності розглянутих методів наближеного розв'язку спектральної крайової задачі. Є також зауваження, щодо недостатньої апроксимації та розміру квадрату, який можна було вибрати більшим, внаслідок чого на графіку та у таблицях розділ між власними значеннями буде більш помітний. Але не вважаю ці зауваження такими, що зменшує загальну позитивну оцінку роботи.

Незважаючи на те, що робота носить обчислювально-експериментальний характер, вона в собі має основу для подальших теоретичних досліджень.

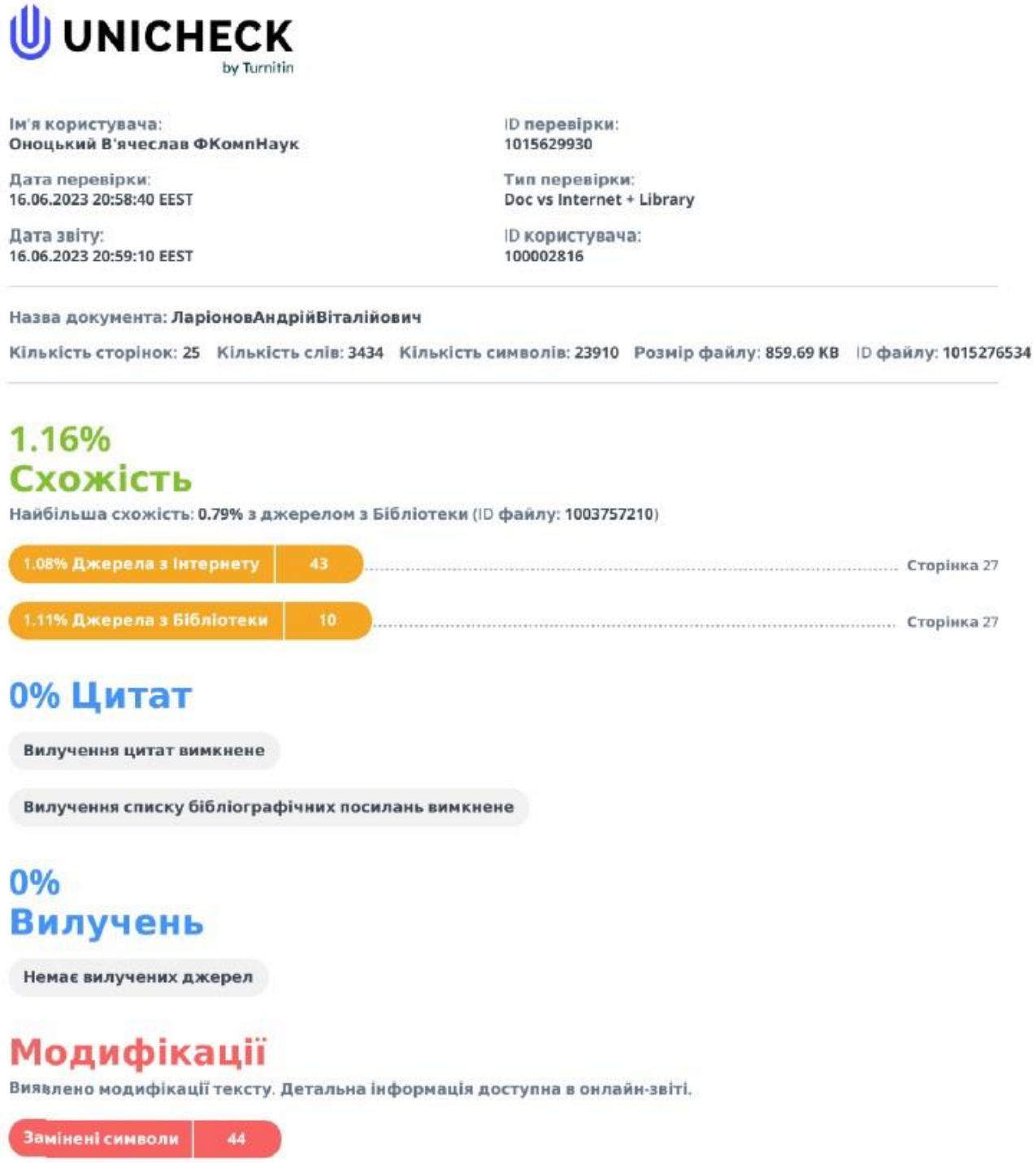
Вважаю, що кваліфікаційна робота студента Андрія Ларіонова відповідає вимогам, які висуваються до бакалаврських робіт, а її автор заслуговує на присвоєння кваліфікації бакалавра. Рекомендую оцінку «добре» (85).

Доцент кафедри моделювання складних систем
факультету комп'ютерних наук та кібернетики
Київського національного університету
імені Тараса Шевченка,
доктор фіз.-мат. наук, доцент Андрій ШАТИРКО

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА

СИСТЕМА ЗАПОБІГАННЯ ТА ВИЯВЛЕННЯ АКАДЕМІЧНОГО ПЛАГІАТУ

Довідка про оригінальність кваліфікаційної роботи за освітнім рівнем бакалавр



Експертна оцінка роботи науковим керівником :

Робота виконана самостійно та не містить відомостей без посилань на джерела.

Науковий керівник: _____ Семенов В.В.

(підпис) (ПІБ)

Оператор: _____ Оноцький В.В.

(підпис) (ПІБ)