НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ

ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики

Кафедра прикладної математики

ЗВІТ

про виконання лабораторної роботи №1

з дисципліни «Чисельні методи математичної фізики»

на тему

«Задачі на граничні умови»

Варіант 1

|  |  |
| --- | --- |
| Виконав: | Перевірив: |
| студент групи КМ-11мн | Проф. |
| Агафонов Д. С. | Ориняк І. В. |

Київ — 2021

ЗМІСТ

[1 ПОСТАНОВКА ЗАДАЧІ 2](#_Toc87211896)

[2 ТЕОРЕТИЧНІ ВІДОМОСТІ 3](#_Toc87211897)

[3 АНАЛІТИЧНИЙ РОЗВ’ЯЗОК 5](#_Toc87211898)

[4 ПРОГРАМНА РЕАЛІЗАЦІЯ 7](#_Toc87211899)

[ВИСНОВКИ 12](#_Toc87211900)

[ДОДАТОК А. Лістинг 13](#_Toc87211901)

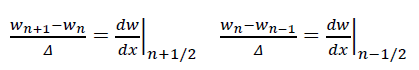
# 1 ПОСТАНОВКА ЗАДАЧІ

Задача дається диференційними рівняннями

Граничні умови такі , . Значення при , значення . Рішити ПЕРШИМ підходом (один параметр -1 диф рівняння 2-го порядку).

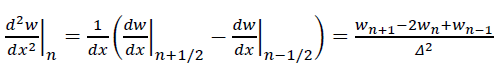
# 2 ТЕОРЕТИЧНІ ВІДОМОСТІ

Як видно з графічного представлення похідної різниця між двома сусідніми значеннями функції, поділена на довжину інтервалу між ними, дає приблизно значення похідної в точці, що розміщена між ними (Формула 2.1).



Формула 2.1

Якщо функція задана в точках розбивки, то перша похідна знаходиться в проміжних точках (Формула 2.2).



Формула 2.2

Згідно загальної схеми вирішення краєвих задач 2-го порядку весь відрізок розбивається на N проміжків. Сумарна кількість всіх вузлових точок (по яким відбувалося розбиття) і двох кінців відрізку рівна N+1, кожна з яких номеруються як 0, 1, 2, 3, ..., N відповідно їхнього порядку. В кожній точці вводиться 1 невідома, якщо вводиться одне рівняння 2-го порядку. В кожній внутрішній точці записується основне різницеве рівняння. Оскільки внутрішніх точок 𝑁−1, то отримуємо 𝑁−1 рівнянь. Вони доповнюються 2 граничними умовами – по одній зліва і справа.

Формулювання задачі (Формула 2.3):



Формула 3

Де 𝑢 - переміщення направлене вздовж осі 𝑥, 𝑁 - сила, яку для зручності направимо в протилежну сторону 𝑥.

Формуючи одне диф рівняння 2го порядку, отримаємо (Формула 2.4):



Формула 2.4

Відповідно до (2.1) різницева схема рішення (2.4), де всі значення, відповідно до загальної схеми беруться в точці має вигляд (Формула 2.5):

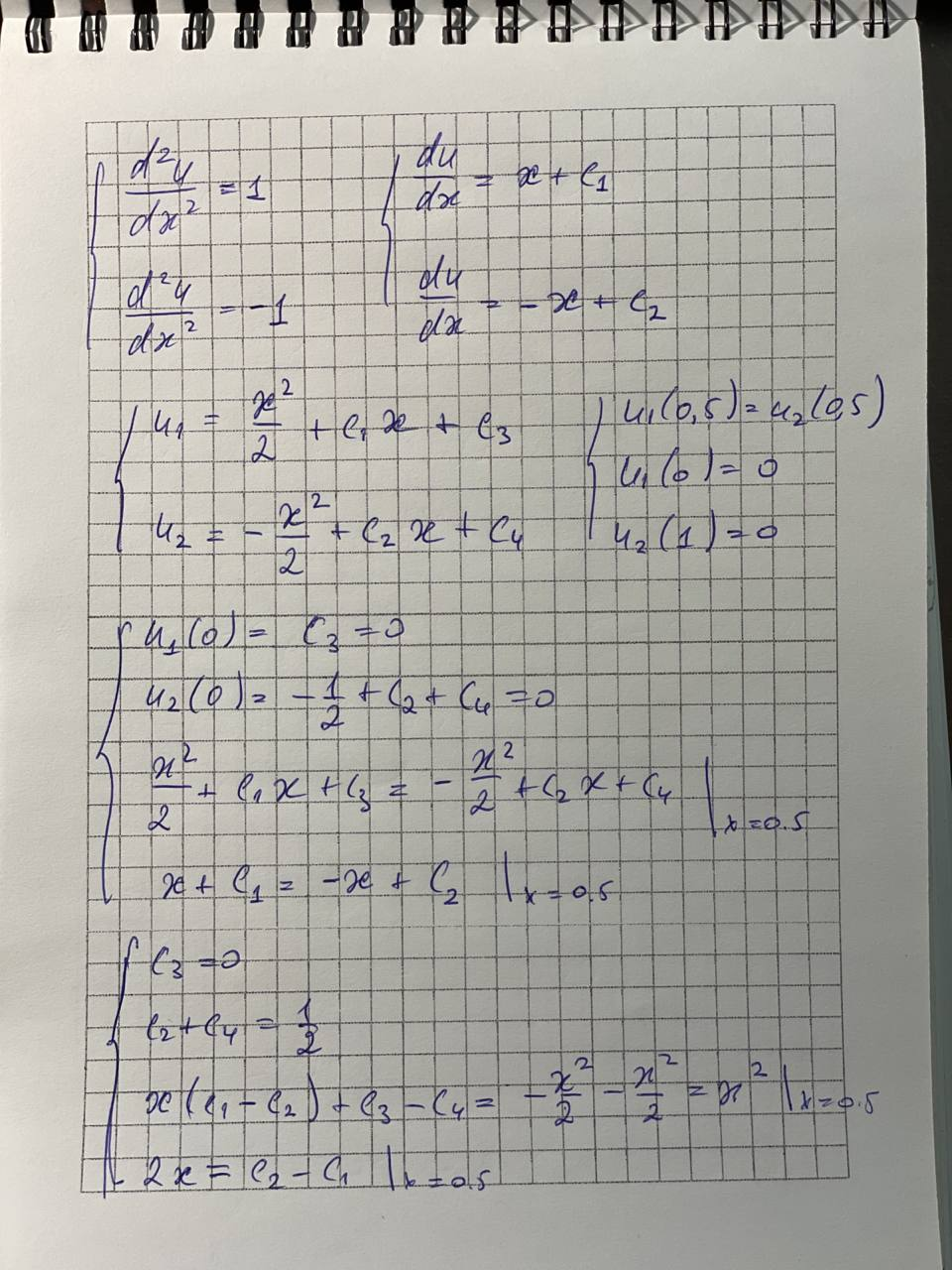


Формула 2.5

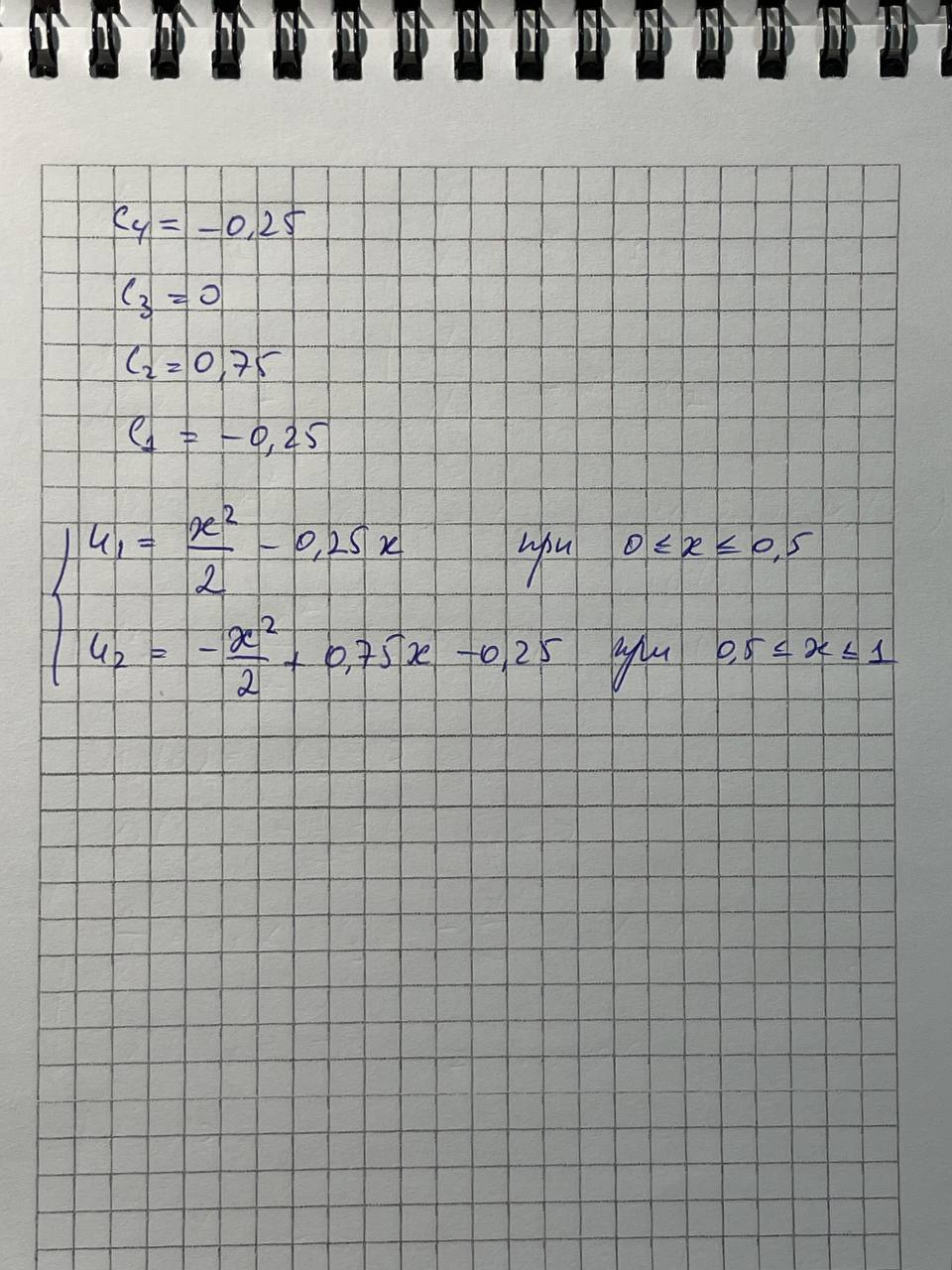
Де -- довжина проміжку між двома сусідніми точками; α(x) -- приймає в залежності від х значення 1 та -1.

# 3 АНАЛІТИЧНИЙ РОЗВ’ЯЗОК

Пошук точного розв’язку системи диференційних рівнянь 1-го порядку (1) будемо розглядати на кожному із проміжків 0 ≤ x ≤ 0.5 та 0.5 ≤ x ≤ 1 окремо, враховуючи розрив в точці x = 0.5. Будемо шукати розв’язок системи наступних диференційних рівнянь 2-го порядку:



В результаті отримаємо коефіцієнти та аналітичне рішення має вигляд:



# 4 ПРОГРАМНА РЕАЛІЗАЦІЯ

Відрізок розбивається на N проміжків. Сумарна кількість всіх вузлових точок (по яким відбувалося розбиття) і двох кінців відрізку рівна N+1, кожна з яких номеруються як 0, 1, 2, 3, ..., N відповідно їхнього порядку. Для пошуку розв’язку (5) складемо систему з N рівнянь та двох граничних умов.

Система має вигляд , де А – матриця з невідомими змінними, b – вектор вільних коефіцієнтів. Зв’язок між сусідніми точками та дві граничні умови мають вигляд:

При розбитті на 4 інтервали, отримуємо такі вузлові точки: [0. , 0.25, 0.5, 0.75, 1.].

Необхідно розв’язати систему рівнянь у матричному вигляді за допомогою функції solve модуля numpy.linalg. Матриця А з граничними умовами (рис. 4.1):

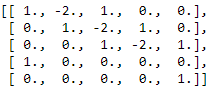


Рис. 4.1 – Матриця А

та вектор b (рис.4.2):



Рис. 4.2 – Матриця b

Розв’язки системи для n=4 в кожному вузлі(рис. 4.3):



Рис. 4.3 - Розв’язки системи

Візуалізація апроксимації для n=4 (рис.4.4):

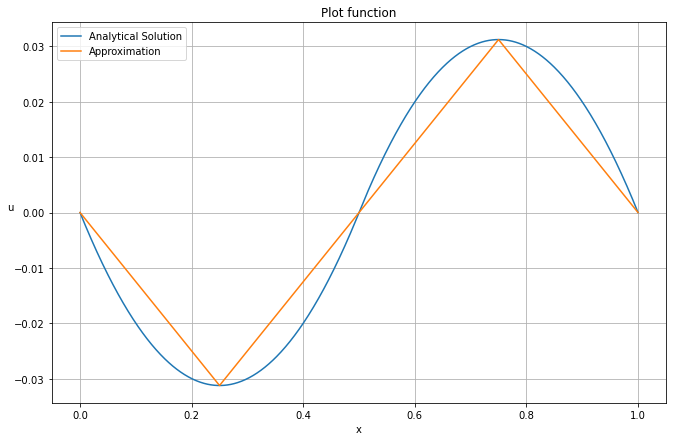


Рис. 4.4 – Апроксимація n=4

Апроксимації також були побудовані для n = [4, 6, 8, 10, 16, 32, 54] (рис. 4.5 – 4.11)

|  |  |
| --- | --- |
| Рис. 4.6. – n = 6 | Рис. 4.7 – n = 8 |
| Рис. 4.8 – n = 10 | Рис. 4.9 – n = 16 |
| Рис. 4.10 – n = 32 | Рис. 4.11 – n = 54 |

Для оцінки точності отриманих результатів необхідно обчислити площу фігури, яка обмежена двома вузловими точками, апроксимованою прямою та дійсною кривою. Задача пошуку площі фігур складається в обчисленні двох визначених на проміжку інтегралів на кожному з розбиттів та знаходженні різниці. Наприклад, для двох перших інтервалів при n=4 (рис. 4.12).

Для різних значень n = [4, 6, 8, 10, 16, 32, 54] розраховані сумарні площі по всіх проміжках та відносні похибки обчислень. Дані представлені в таблиці (рис. 4.13).

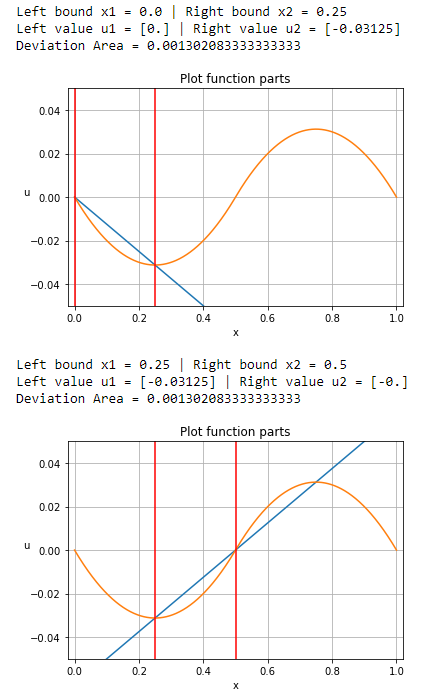


Рис. 4.12 – Розрахунки та візуалізація оцінювання точності n=4

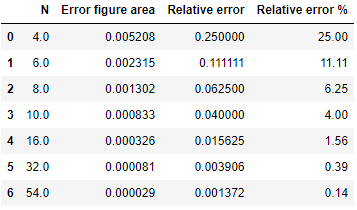


Рис. 4.13 – Таблиця оцінки точності

# ВИСНОВКИ

В ході виконання лабораторної роботи були проведені розрахунки для отримання аналітичного розв’язку та створений код для програмної реалізації розв’язання задачі. Було знайдено чисельнимим методами невідому функцію та похибку обчислень для індивідуального завдання. Результати роботи представлені в табличній і графічній формі. Наведено результати для різної щільності розбиття відрізку.

Для гарантування заданої точності 3-5% можна обрати крок n=10, оскільки відносна похибка обчислень для обраного n дорівнює 4%.

# ДОДАТОК А. Лістинг

import numpy as np

import pandas as pd

from scipy import integrate

impоrt matplotlib.pyplot as plt

### Initial Conditions

x\_0, x\_n = 0, 1

u\_0, u\_n = 0, 0

N = 4

delta = (x\_n - x\_0) / N

x = np.linspace(x\_0, x\_n, N+1)

x

### Analytical Solution

xx = np.linspace(x\_0, x\_n, 1000)

uu\_f = np.vectorize(lambda x: x\*\*2 / 2 - 0.25\*x if x <= 0.5 else -x\*\*2 / 2 + 0.75\*x - 0.25 )

uu = uu\_f(xx)

plt.figure(1, figsize=(9, 6))

plt.plot(xx, uu, label='Analytical Solution')

plt.xlabel('x')

plt.ylabel('u', rotation=0)

plt.title('Analytical Solution')

plt.grid(True)

plt.show(1)

### First Approach

#### Construct matrix

def constr\_matrix(N):

A = np.zeros((N+1, N+1))

np.fill\_diagonal(A[:N, :N-1] , 1)

np.fill\_diagonal(A[:N, 1:N], -2)

np.fill\_diagonal(A[:N, 2:N+1], 1)

A[N-1, 0], A[N, N] = 1, 1

return A

A = constr\_matrix(N)

A

#### Construct vector

def constr\_vect(N, delta, x, u\_0, u\_n):

b = np.zeros((N+1, 1))

f\_b = np.vectorize(lambda x: 1 if 0 <= x < 0.5 else (0 if x == 0.5 else -1))

b[:-2] = (delta\*\*2)\*f\_b(x[1:-1]).reshape(-1,1)

b[-2], b[-1] = u\_0, u\_n

return b

b = constr\_vect(N, delta, x, u\_0, u\_n)

b

#### Solve System

result = np.linalg.solve(A, b)

result

### Plot Result

def plot\_result(x, result, xx=xx, uu=uu):

plt.figure(2, figsize=(11, 7))

plt.plot(xx, uu, label='Analytical Solution')

plt.plot(x, result, label='Approximation')

plt.xlabel('x')

plt.ylabel('u', rotation=0)

plt.title('Plot function')

plt.legend()

plt.grid(True)

plt.show(2)

plot\_result(x, result, xx=xx, uu=uu)

N = 54

delta = (x\_n - x\_0) / N

x10 = np.linspace(x\_0, x\_n, N+1)

x10

A = constr\_matrix(N)

b = constr\_vect(N, delta, x10, u\_0, u\_n)

result10 = np.linalg.solve(A, b)

plot\_result(x10, result10, xx=xx, uu=uu)

### Calculate Error

N = 4

delta = (x\_n - x\_0) / N

x = np.linspace(x\_0, x\_n, N+1)

A = constr\_matrix(N)

b = constr\_vect(N, delta, x, u\_0, u\_n)

u\_i = np.linalg.solve(A, b)

def line\_from\_points(x, x1, y1, x2, y2):

return (x - x1) \* (y2 - y1) / (x2 - x1) + y1

def calc\_error(x, u\_i, verb=True):

error = 0

for x1, x2, u1, u2 in zip(x[:-1], x[1:],

u\_i[:-1], u\_i[1:]):

val1, \_ = integrate.quad(uu\_f, a=x1, b=x2)

val2, \_ = integrate.quad(line\_from\_points, x1, x2,

args=(x1, u1, x2, u2))

error += np.abs(val1-val2)

if verb==True:

print(f'Left bound x1 = {x1}', f'| Right bound x2 = {x2}')

print(f'Left value u1 = {u1}', f'| Right value u2 = {u2}')

# print(f'Error = {np.abs(val1-val2)}')

# print(point2x\_line(x, x1, u1, x2, u2))

print(f'Deviation Area = {np.abs(val1-val2)}')

plt.plot(x, line\_from\_points(x, x1, u1, x2, u2))

plt.plot(xx, uu)

plt.axvline(x1, c='r')

plt.axvline(x2, c='r')

plt.xlim(-0.02, 1.02)

plt.ylim(-0.05, 0.05)

plt.xlabel('x')

plt.ylabel('u', rotation=0)

plt.title('Plot function parts')

plt.grid(True)

plt.show()

return error

error = calc\_error(x, u\_i)

print(f'Summary area = {error:5f}')

total\_area = (abs(integrate.quad(uu\_f, x\_0, 0.5)[0]) + abs(integrate.quad(uu\_f, 0.5, x\_n)[0]))

print(f'Total figure area = {total\_area:6f}')

relat\_error = error / total\_area

print(f'Relative Error = {relat\_error:6f}')

error\_df = pd.DataFrame(columns=['N', 'Error figure area', 'Relative error'])

N\_array = np.array([4, 6, 8, 10, 16, 32, 54])

for idx, N in enumerate(N\_array):

delta = (x\_n - x\_0) / N

x = np.linspace(x\_0, x\_n, N+1)

A = constr\_matrix(N)

b = constr\_vect(N, delta, x, u\_0, u\_n)

u\_i = np.linalg.solve(A, b)

error = calc\_error(x, u\_i, verb=False)

relat\_error = error / total\_area

error\_df.loc[idx] = N, error, relat\_error

error\_df['Relative error %'] = np.around(error\_df['Relative error']\*100, 2)

error\_df