НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ

ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики

Кафедра прикладної математики

Звіт

із лабораторної роботи №2  
з дисципліни «Нелінійний аналіз»

|  |  |
| --- | --- |
| Виконав: | Перевірив: |
| студент групи КМ-52 | Сірик С.В. |
| Сєльський Є. П. |
|  |  |
|  |  |
|  |  |

Київ — 2018

Зміст

[1 Теоретичні Відомості 3](#_Toc532547385)

[1.1 Фрактал Мандельброта 3](#_Toc532547386)

[1.2 Фрактал Жюліа 3](#_Toc532547387)

[2 Результати Роботи Програми 4](#_Toc532547388)

[Висновки 13](#_Toc532547389)

[Додаток А](#_Toc532547390) [Лістинг Програм 14](#_Toc532547391)

# 1 ТЕОРЕТИЧНІ ВІДОМОСТІ

Фрактал — нерегулярна, самоподібна структура. В широкому розумінні фрактал означає фігуру, малі частини якої в довільному збільшенні є подібними до неї самої.

## 1.1 Фрактал Мандельброта

Множина Мандельброта — обмежена та зв’язна множина на комплексній площині, межа якої утворює фрактал.  
Сукупність елементів С поля комплексних чисел, для яких послідовність

{zn : n ≥ 0}, що визначена ітераційно за правилом

*,* дезадовольняє умову sup| zn | < ∞ називається множиною Мандельброта.

## 1.2 Фрактал Жюліа

Розглянемо функцію f:C→C. Множина Жюліа визначається як границя множини точок Z, які прямують до нескінченності при інтегруванні f(z): .

# 2 РЕЗУЛЬТАТИ РОБОТИ ПРОГРАМИ

# 

На рисунках 2.1 – 2.5 зображені фрактали Мандельброта для функцій , , , , відповідно.

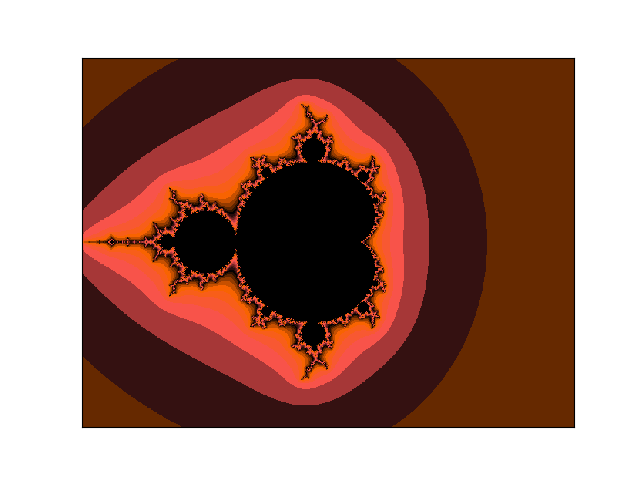
**

Рис. 2.1. Фрактал Мандельброта функції

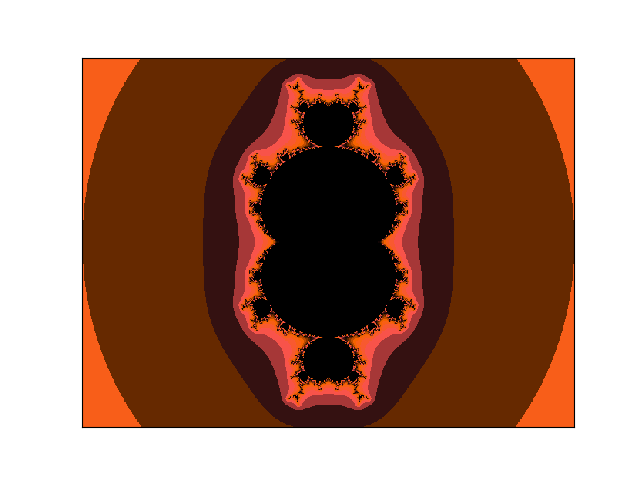
**

Рис. 2.2. Фрактал Мандельброта функції

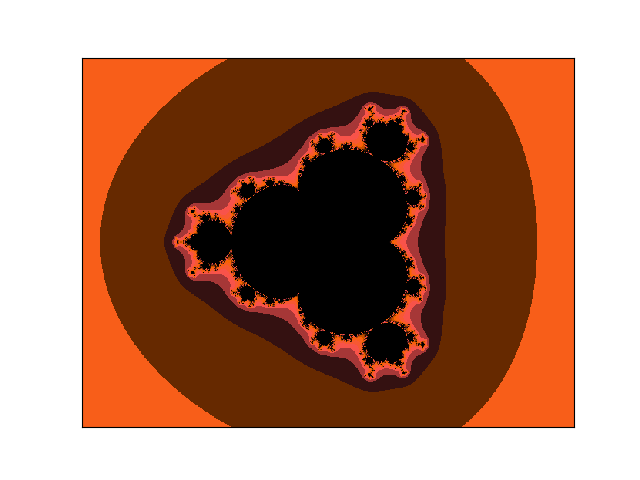
**

Рис. 2.3. Фрактал Мандельброта функції

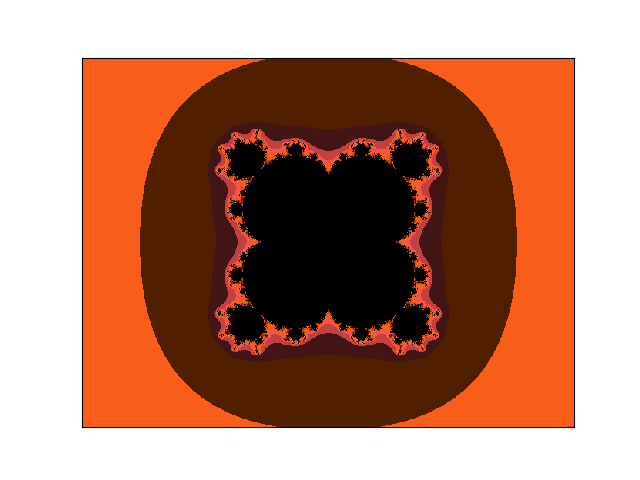
**

Рис. 2.4. Фрактал Мандельброта функції

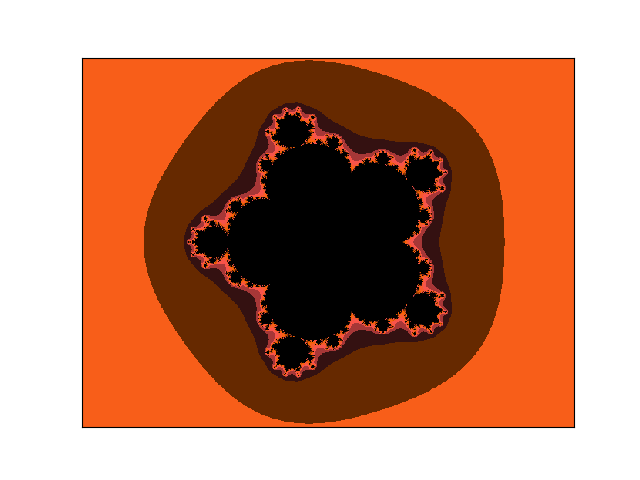
**

Рис. 2.5. Фрактал Мандельброта функції

На рисунках 2.6 – 2.7 зображені фрактали Мандельброта для функцій , відповідно.

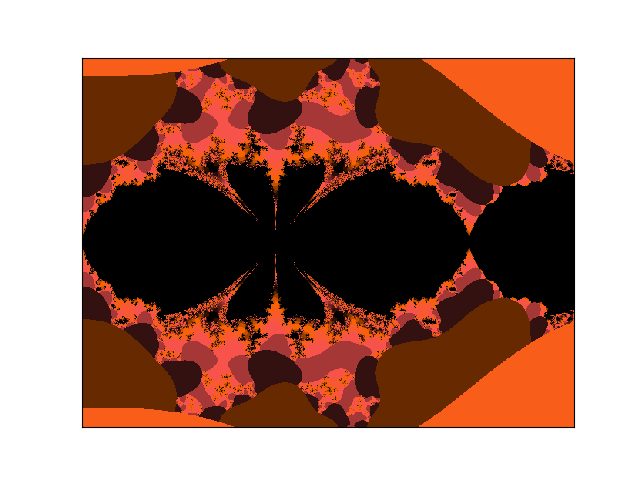
**

Рис. 2.6. Фрактал Мандельброта функції

**

Рис. 2.7. Фрактал Мандельброта функції

На рисунках 2.8 – 2.16 зображені фрактали Жюліа для степеневих функцій з різними значеннями константи (за замовчуванням ).

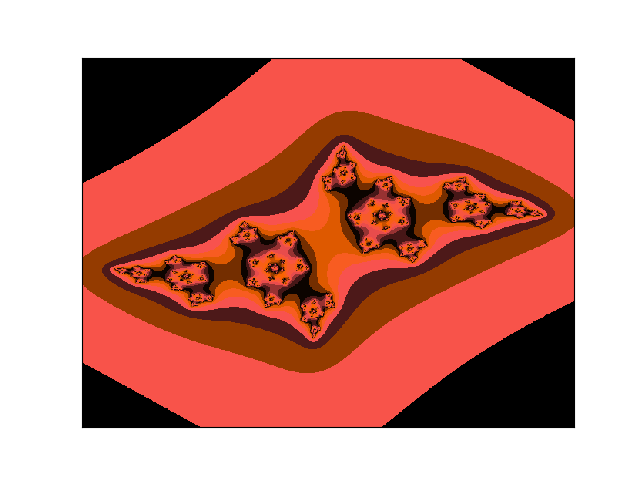


Рис. 2.8. Фрактал Жюліа функції

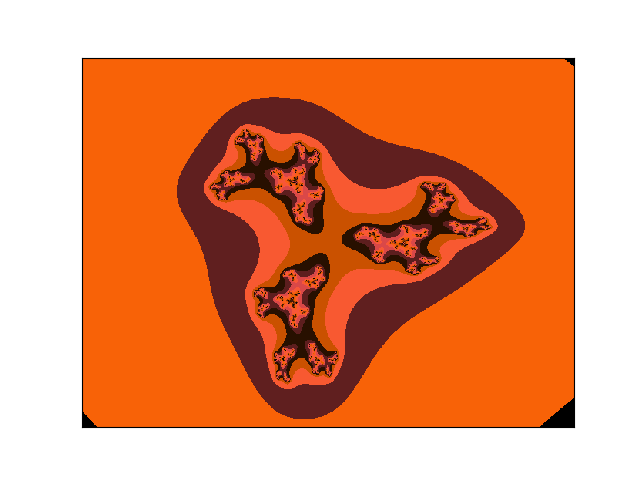


Рис. 2.9. Фрактал Жюліа функції



Рис. 2.10. Фрактал Жюліа функції

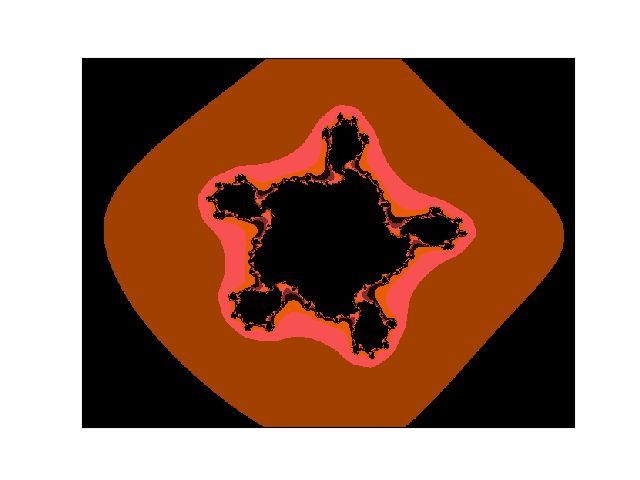


Рис. 2.11. Фрактал Жюліа функції

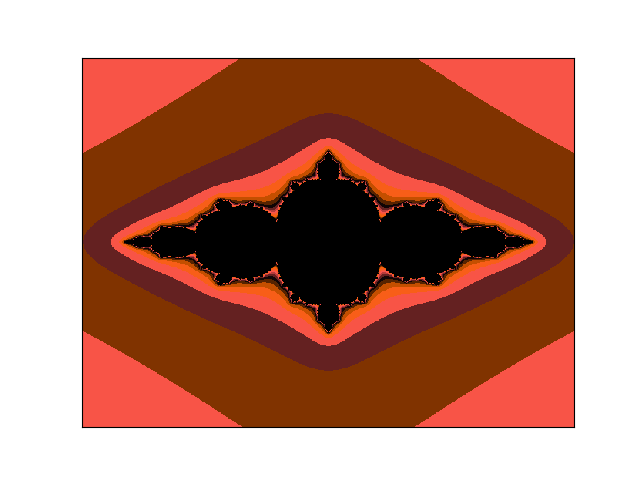


Рис. 2.12. Фрактал Жюліа функції ,

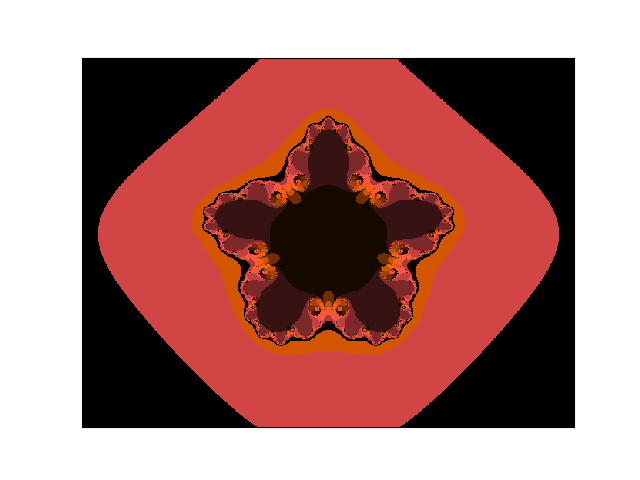


Рис. 2.13. Фрактал Жюліа функції ,

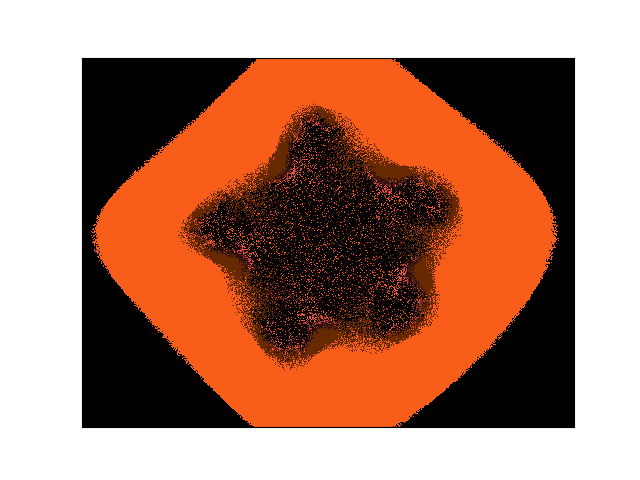


Рис. 2.14. Фрактал Жюліа функції ,

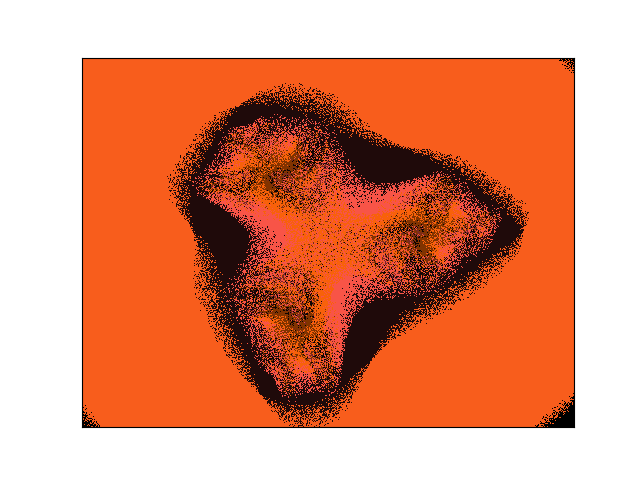


Рис. 2.15. Фрактал Жюліа функції ,

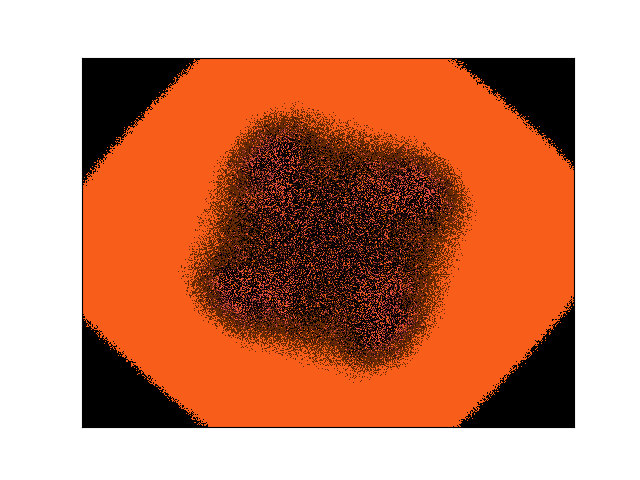


Рис. 2.16. Фрактал Жюліа функції ,

# ВИСНОВКИ

Під час виконання даної лабораторної роботи було написане програмне забезпечення мовою програмування python, яке дозволяє будувати растрові зображення фракталів Мандельброта та Жюліа для різних наборів вхідних параметрів: кількості точок по осям, початкових та кінцевих точок, кількості ітерацій, виду рекурентного співвідношення, виду константи , граничного значення, з яким порівнюється модуль числа на кожній ітерації.

Для фракталів Жюліа змодельовані різні випадки задання :

* При постійній фрактали мають очікуваний вигляд і представляють собою частинки відповідних фракталів Мандельброта;
* При постійній з нульовою уявною частиною фрактал схожий на фрактал Мандельброта і стає симетричним відносно початку координат;
* При постійній з нульовою дійсною частиною фрактал стає симетричним відносно осі ;
* При випадковому заданні значень дійсної та уявної частин числа зображення фракталу «розсіюється» ближче до центру.

# Додаток А

# Лістинг програм

Лістинг файлу fractals.py:

import numpy as np

from itertools import cycle

import matplotlib.pyplot as plt

import math

import matplotlib.colors as clr

import random

def mandelbrot\_set(iterations, top\_border, x\_min, x\_max, y\_min, y\_max, x\_points, y\_points, deg\_z):

portrait = np.zeros([x\_points, y\_points])

for i, x in enumerate(np.linspace(x\_min, x\_max, x\_points)):

for j, y in enumerate(np.linspace(y\_min, y\_max, y\_points)):

c = x + 1j \* y

z = 0

for k in range(iterations):

z = z \*\* deg\_z + c

if abs(z) > top\_border:

portrait[i, j] = k

break

colorpoints = [(1 - (1 - q) \*\* 4, c) for q, c in zip(np.linspace(0, 1, 20), cycle(['#f85252', '#000000', '#f96400',]))]

cmap = clr.LinearSegmentedColormap.from\_list('mycmap', colorpoints, N=2048)

fig = plt.figure()

fig.add\_subplot()

plt.xticks([])

plt.yticks([])

plt.imshow(-portrait.T, cmap=cmap, interpolation='none')

return portrait

def mandelbrot\_set\_trig(iterations, top\_border, x\_min, x\_max, y\_min, y\_max, x\_points, y\_points, function):

portrait = np.zeros([x\_points, y\_points])

for i, x in enumerate(np.linspace(x\_min, x\_max, x\_points)):

for j, y in enumerate(np.linspace(y\_min, y\_max, y\_points)):

c = x + 1j \* y

z = 0

for k in range(iterations):

z = function(z) + c

if abs(z) > top\_border:

portrait[i, j] = k

break

colorpoints = [(1 - (1 - q) \*\* 4, c) for q, c in zip(np.linspace(0, 1, 20), cycle(['#f85252', '#000000', '#f96400',]))]

cmap = clr.LinearSegmentedColormap.from\_list('mycmap', colorpoints, N=2048)

fig = plt.figure()

fig.add\_subplot()

plt.xticks([])

plt.yticks([])

plt.imshow(-portrait.T, cmap=cmap, interpolation='none')

return portrait

def julia\_set(iterations, top\_border, x\_min, x\_max, y\_min, y\_max, x\_points, y\_points, deg\_z, c\_Re, c\_Im):

portrait = np.zeros([x\_points, y\_points])

for i, x in enumerate(np.linspace(x\_min, x\_max, x\_points)):

for j, y in enumerate(np.linspace(y\_min, y\_max, y\_points)):

if c\_Re is None and c\_Im is None:

c = random.random() + 1j \* random.random()

elif c\_Re is None and c\_Im is not None:

c = random.random() + 1j \* c\_Im

elif c\_Im is None and c\_Re is not None:

c = c\_Re + 1j \* random.random()

else:

c = c\_Re + 1j \* c\_Im

z = math.asinh(x) + 1j \* math.asinh(y)

for k in range(iterations):

z = z \*\* deg\_z + c

if abs(z) > top\_border:

portrait[i, j] = k

break

colorpoints = [(1 - (1 - q) \*\* 4, c) for q, c in

zip(np.linspace(0, 1, 20), cycle(['#f85252', '#000000', '#f96400', ]))]

cmap = clr.LinearSegmentedColormap.from\_list('mycmap', colorpoints, N=2048)

fig = plt.figure()

fig.add\_subplot()

plt.xticks([])

plt.yticks([])

plt.imshow(-portrait.T, cmap=cmap, interpolation='none')

return portrait