Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Кафедра прикладної математики

Звіт
про виконання лабораторної роботи
з дисципліни “Основи нелінійного аналізу”

Студента групи КМ-71
Бірук С.В.

Київ – 2020

ЗМІСТ

1 ПОСТАНОВКА ЗАДАЧІ

Розглянути та дослідити поведінку множин Жюліа для різних значень комплексної константи. Визначити тип нерухомих точок. Зобразити отримані результати в декількох кольорових гаммах.

2 ТЕОРЕТИЧНІ ВІДОМОСТІ

2.1 Ітерації комплексних функцій. Множини Жюліа

Множини Жюліа дають найбільш вражаючі ілюстрації того, як простий процес може призвести до побудови надзвичайно складних множин. В межах даної роботи будемо розглядати ітерації наступного вигляду: $z_{n+1} = z_n^2 + c$, де c – комплексна не нульова константа.

Множини Жюліа появляються в результаті ітерації функції f комплексної змінної z і відносяться до дискретних динамічних систем. В загальному вигляді, множини Жюліа – це динамічний репеллер (відштовхуюча множина). Як правило – фрактал.

Для функцій, які є аналітичними на комплексній множині, ми можемо використовувати потужні методи теорії функції комплексної змінної для отримання більш детальної інформації про структуру аналогічних відштовхуючих множин.

2.2 Основи теорії множин Жюліа

Для зручності викладення інформації будемо припускати, що $f: C \to C$ є поліномом степені $n \geq 2$ з комплексними коефіцієнтами, $f(z) = a_0 + a_1 z + \ldots + a_n z^n$. Дамо короткий опис основних понять теорії комплексних відображень.

Будемо позначати через f^k k-ту ітерацію (композицію) функції f. Якщо $f(w) = w$, то точку w назвемо нерухомою точкою f, і якщо $f^p(w) = w$, для деякого цілого числа p > 1 – то періодичною точкою f; найменше p, для якого виконується рівність $f^p(w) = w$ називається періодом w. Ми будемо називати $w, f(w), \ldots, f^p(w)$ орбітою періода p. Нехай w – періодична точка періоду p, з похідною $(f^p)'(w) = \lambda$ Тоді w називається:

- притягуючою, якщо $0 \leq |\lambda| < 1$;
- нейтральною, якщо $|\lambda| = 1$;
- відштовхуючою, якщо $|\lambda| > 1$

Сама множина Жюліа функції *f*, що позначається *J(f)*, визначається наступним чином:

$$J(f)=\partial\{z:\lim_{n\to\infty} f^n(z)=\infty\}$$

Таким чином, множина Жюліа функції *f* є границя множини точки *z*, що прагнуть до нескінченності при ітерування *f(z)*.

Найпростіша множина Жюліа відповідає випадку $f(z) = z^2$. Так як $f^{(n)} = z^{(2n)}$, то $f^n(z)\to\infty$тоді і тільки тоді, коли |z| > 1. Границею цієї множини, тобто множиною Жюліа, є одиночне коло $\partial\{z:|z|=1\}$, яке в даному випадку не є фракталом.

Можна переконатися, що множини Жюліа володіють великою різноманітністю. Дійсно, для кожного нового значення *c* можна отримати вражаючі різні зображення.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ

Для подальших обрахунків будемо використовувати наступні бібліотеки:

- numpy — для векторизації обчислень
- sympy — для пошуку нерухомих точок, а також обчислення похідних
- matplotlib — для візуалізації результатів

Пошук нерухомих точок програмним шляхом зображено на рис. 3.1

```
def inform(c):

    z = sympy.symbols('z', imaginary=True)

    eq = sympy.Eq(z, z ** 2 + c)
    sols = sympy.solveset(eq, z)

    for i, zi in enumerate(sols):
        print('Point №{}: {}'.format(i, zi))

        zmodule = abs(sympy.diff(z**2 + c, z).subs(z, zi))
        if zmodule < 1:
            type_point = 'притягуюча'
        elif zmodule > 1:
            type_point = 'відштовхуюча'
        else:
            type_point = 'нейтральна'

        print('Характер нерухомої точки: {}\n'.format(type_point))
```

Рис. 3.1 Пошук нерухомих точок

Дана функція на вхід приймає комплексну константу c. Вирішивши корені рівння, виводиться повідомлення про характер нерухомої точки.

Наступна частина програмної реалізації відображає безпосередньо структуру множини Жюліа.

```
def julia(c, cxpoint = 0, cypoint = 0, zoom = 1, xpoints = 2000,
          ypoints = 2000, max_iterations = 50, infinity_border=4, use_k

    xmin = cxpoint - 1.5 * zoom
    xmax = cxpoint + 1.5 * zoom

    ymin = cypoint - 1.5 * zoom
    ymax = cypoint + 1.5 * zoom

    image = np.zeros((xpoints, ypoints))
    x, y = np.mgrid[xmin:xmax:(xpoints*1j), ymin:ymax:(ypoints*1j)]

    z = x + 1j * y
    for k in range(max_iterations):
        z = z**2 + c
        mask = (np.abs(z) > infinity_border) & (image == 0)
        image[mask] = k if use_k else 1
        z[mask] = np.nan

    return -image.T
```

Рис. 3.2 Реалізація алгоритму Жюліа

Наведена вище функція на вхід приймає комплексну константу, центральну точку відносно якої будуть робитися обчислення, коефіцієнт зумування та кількість ітерацій. Після проведення ітераційного процесу для кожного числа z відносимо його до класу обмеженої або ж необмеженої множини.

Побудуємо декількі ілюстрацій для різних значень комплексної константи, щоб усвідомити наскільки сильно відрізняється оде представлення від іншого при цьому використовуючи одну і ту ж схему ітерацій.

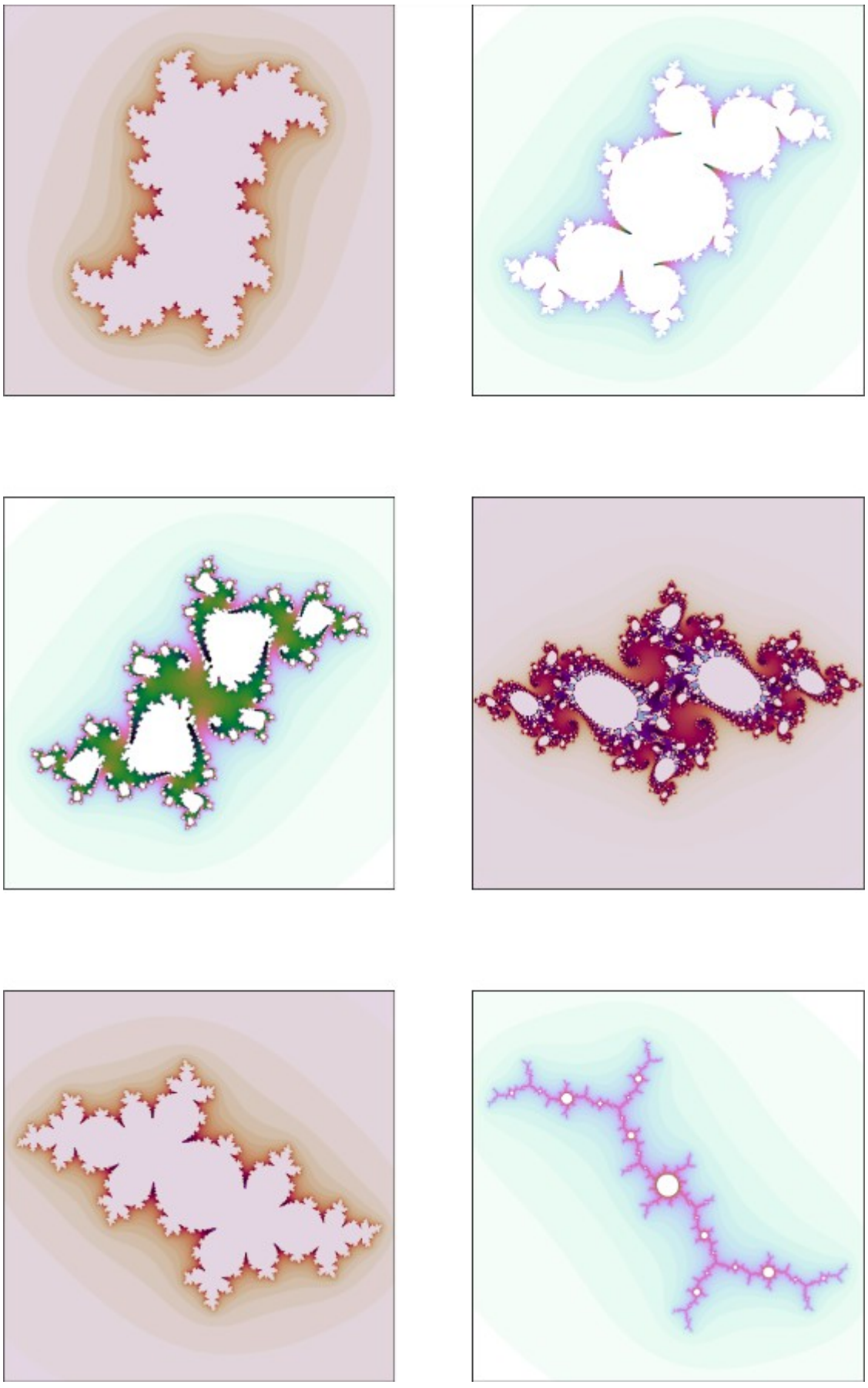

Рис. 3.3 Множини Жюліа для різних значень *c*

Розглянемо більш детально деякі представлення, які найкраще об’яснюють суть множин Жюліа зі сторони естетики і природи фракталів. Але спершу хочу звернути увагу на залежність типу нерухомих точок (атракторів) від вибраної колірної карти.

```
Point №0: -0.142051195135111 - 0.522053385368225*I
Характер нерухомої точки: відштовхуюча

Point №1: 1.14205119513511 + 0.522053385368225*I
Характер нерухомої точки: відштовхуюча


<matplotlib.image.AxesImage at 0x7fe97f8fe520>
```

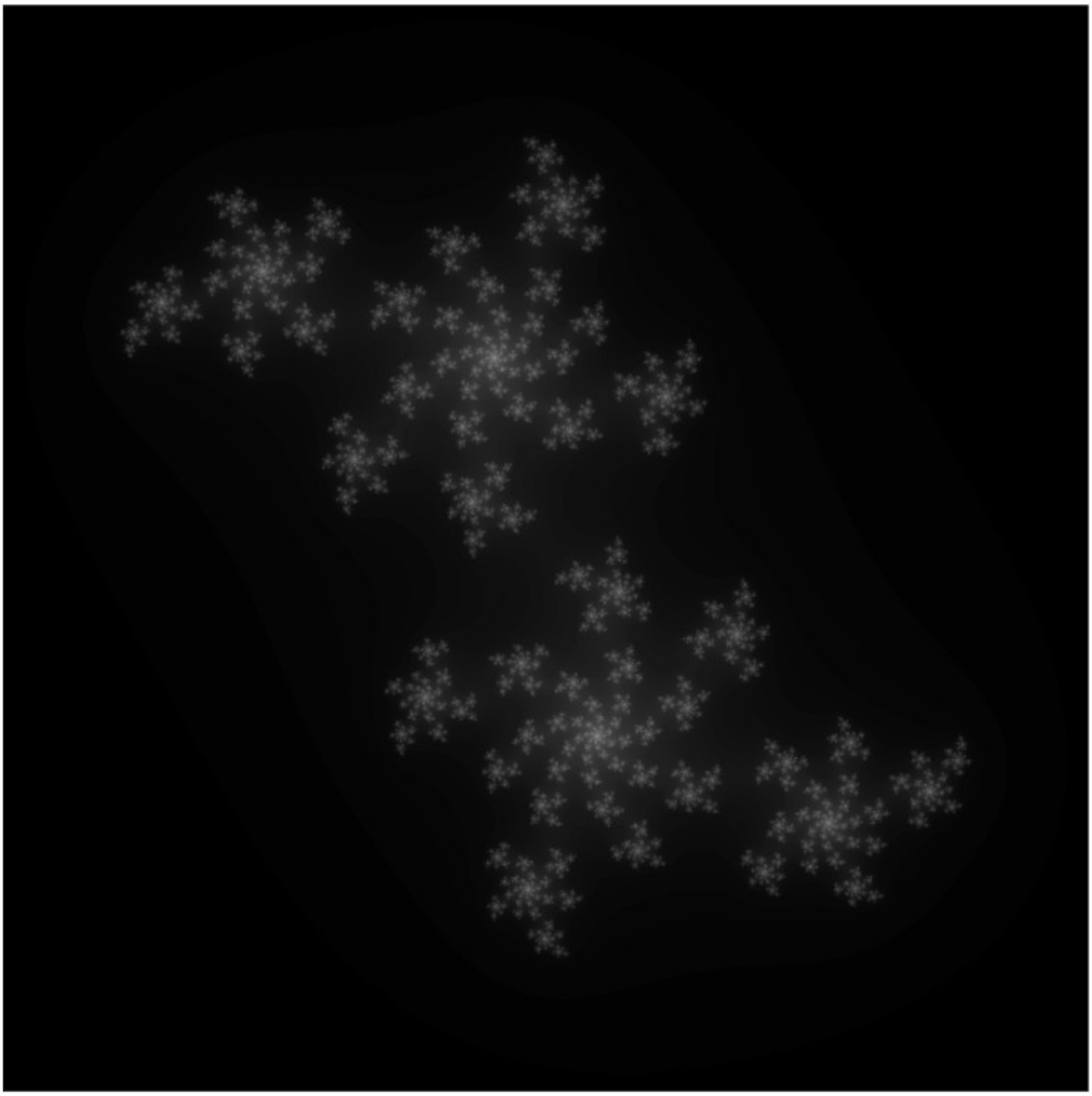

Рис. 3.4 Характер нерухомих точок

При побудові даної ілюстрації було використано наступну колірну карту

Рис. 3.5 Колірна карта (cmap)

Можна зробити висновок, що характер нерухомих точок прямо пропорційний до полюсів кольорової карти. Іншими словами, якщо представити, що злівого полюсу знаходиться відштовчуюча точка то зправа притягуюча. Тому в подальшому, до кожної ілюстрації буде додаватися зображення колірної карти.

Розглянемо більш детально множини Жюліа зі сторони особливостей фракталів.

Рис. 3.6 cubehelix

Ілюстрація на рис. 3.7 яскраво підкреслює самоповторюючу структуру фрактала. З вище описаних припущеннях, ми можемо стверджувати, що центри двох великих спіралей є притягуючими точками. Менші спіралі, які знаходяться навколо головних мають аналогічний характер поведінки своїх центральних точок.

На рис. 3.8 наведено частину вітки спіралі в околі точки 0.53i, де чітко видно повторюваність елементів.

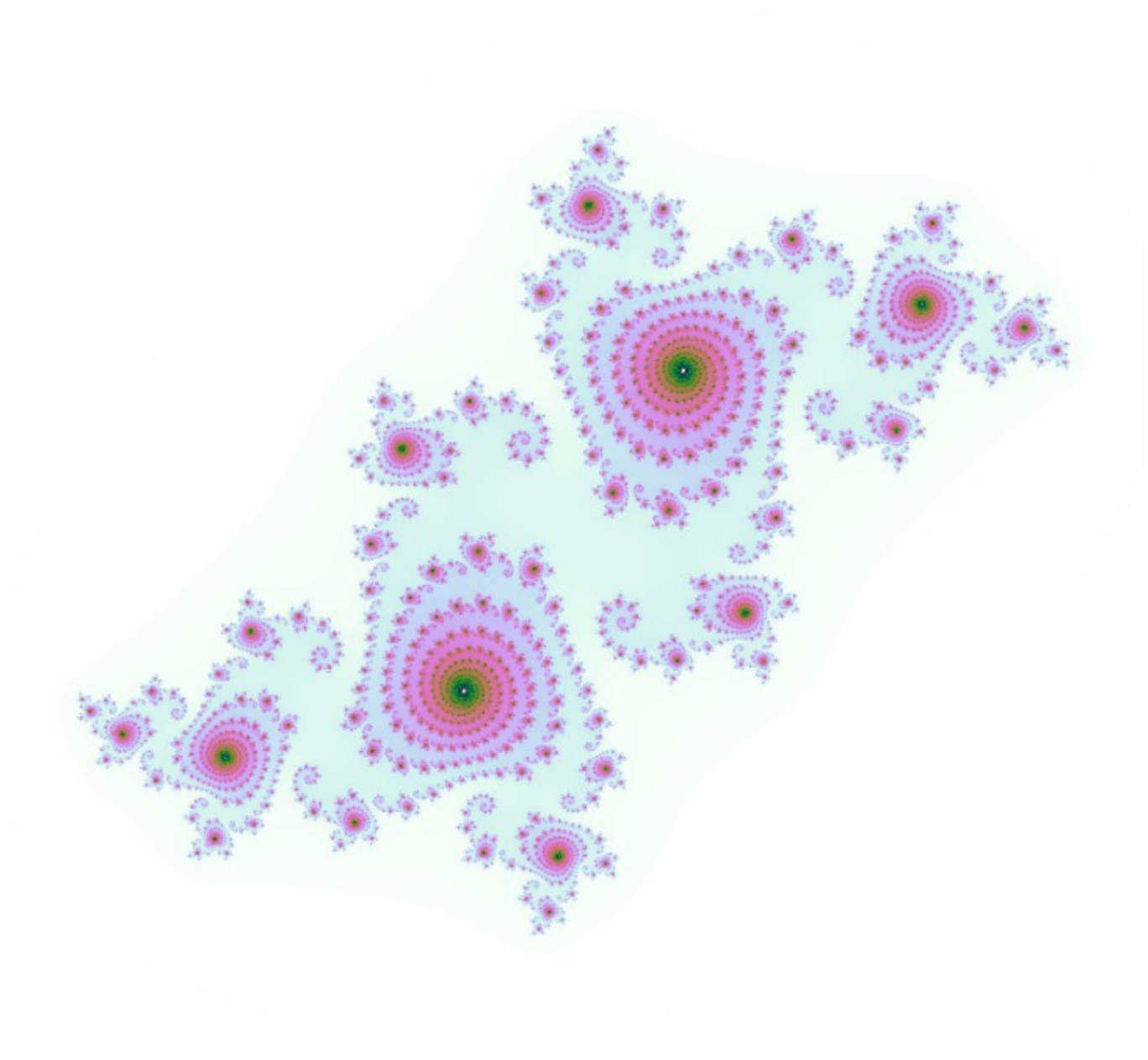

Рис. 3.7 Множина Жюліа для c = -0.194 + 0.6557i

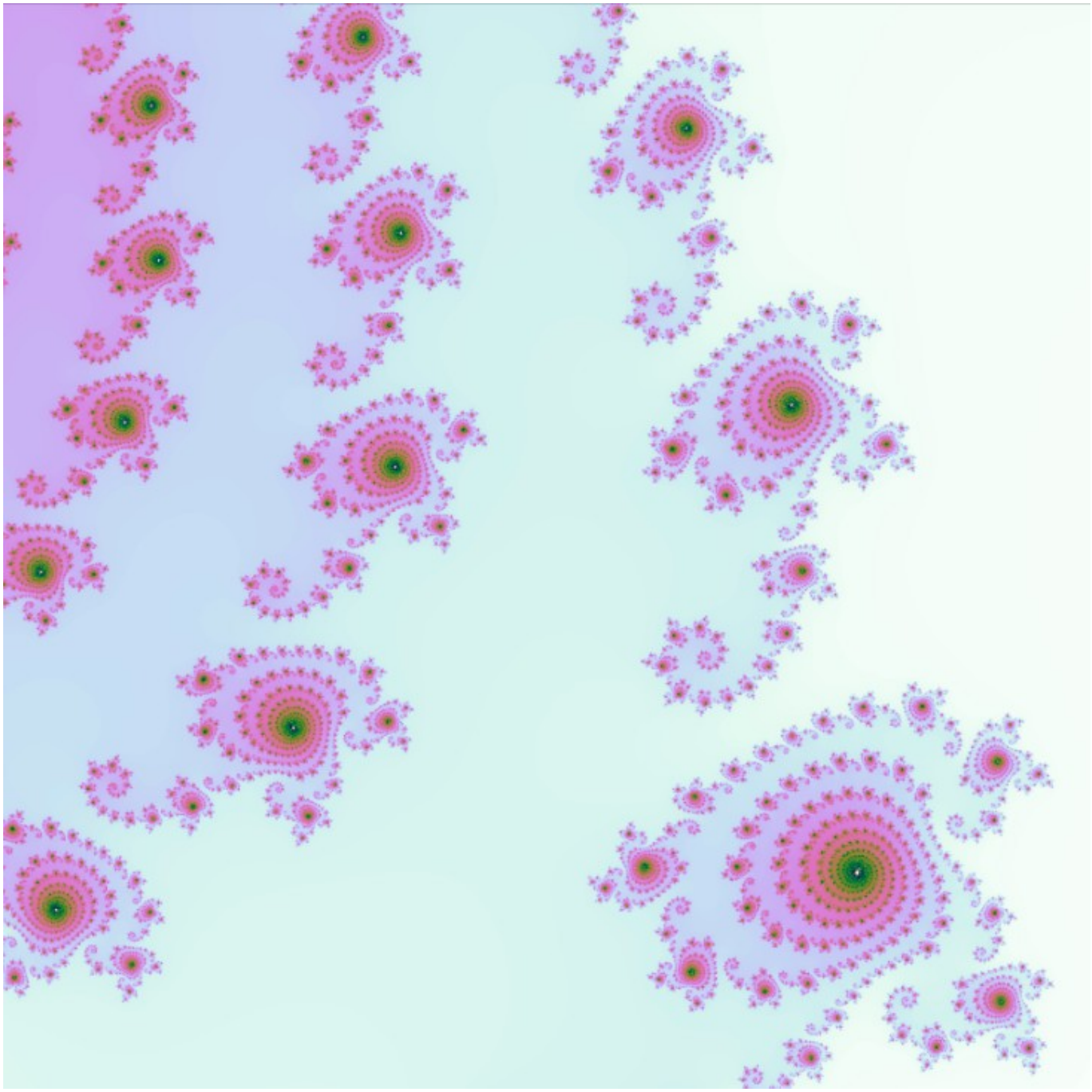

Рис. 3.8 Множина Жюліа для c = -0.194 + 0.6557i в околі точки 0.53i

Наведемо іще декілька яскравних прикладів. Рис. 3.12 зображений в чорнобілому форматі. Це спричиняє ряд недоліків, тому що ми не може визначити напрям збіжності, а також характер нерухомих точок. Однак, естетично виклядає дуже привабливо.

Рис. 3.9 twilight

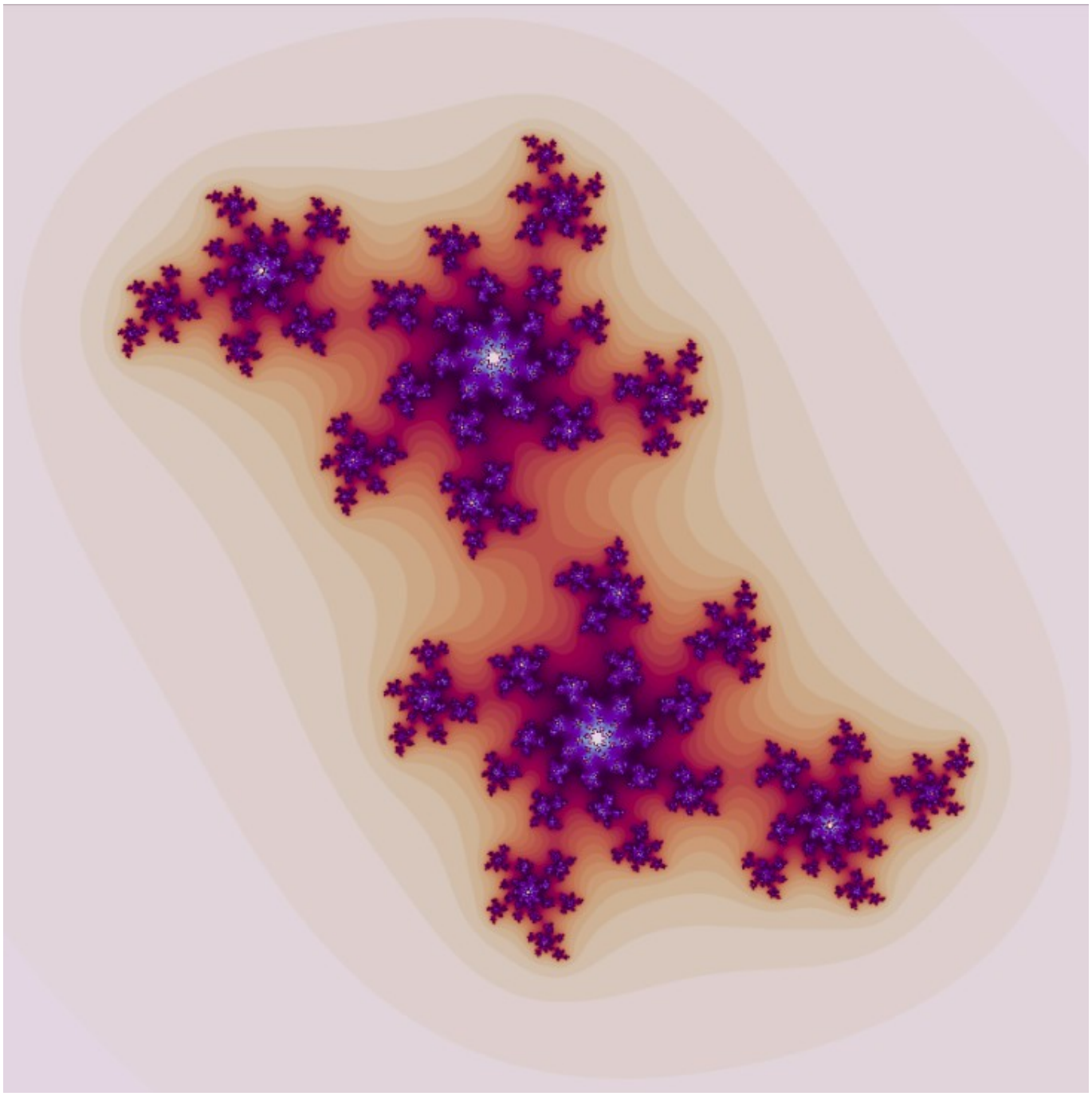

Рис. 3.10 Множина Жюліа для c = 0.11031 - 0.67037i

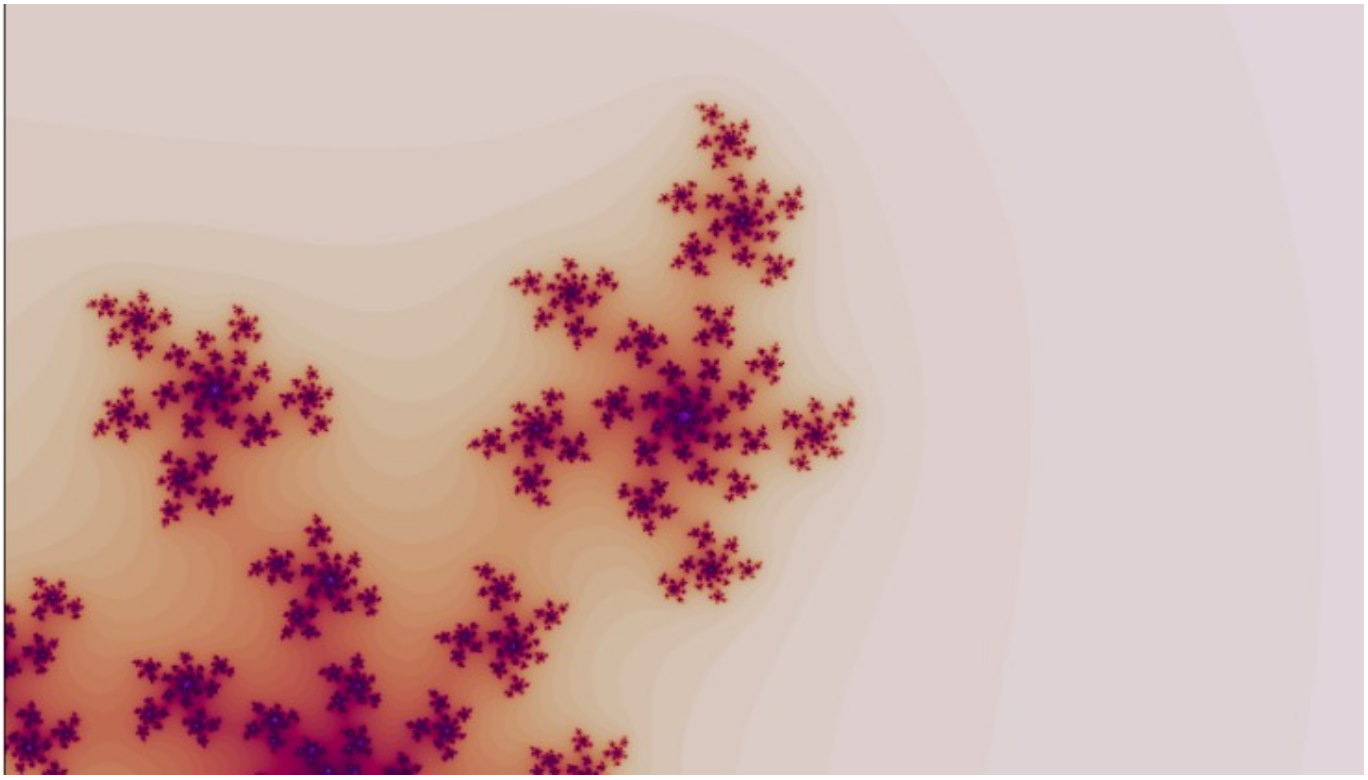

Рис. 3.11 Множина Жюліа для c = 0.11031 - 0.67037i в околі точки 1.14 + 0.52i

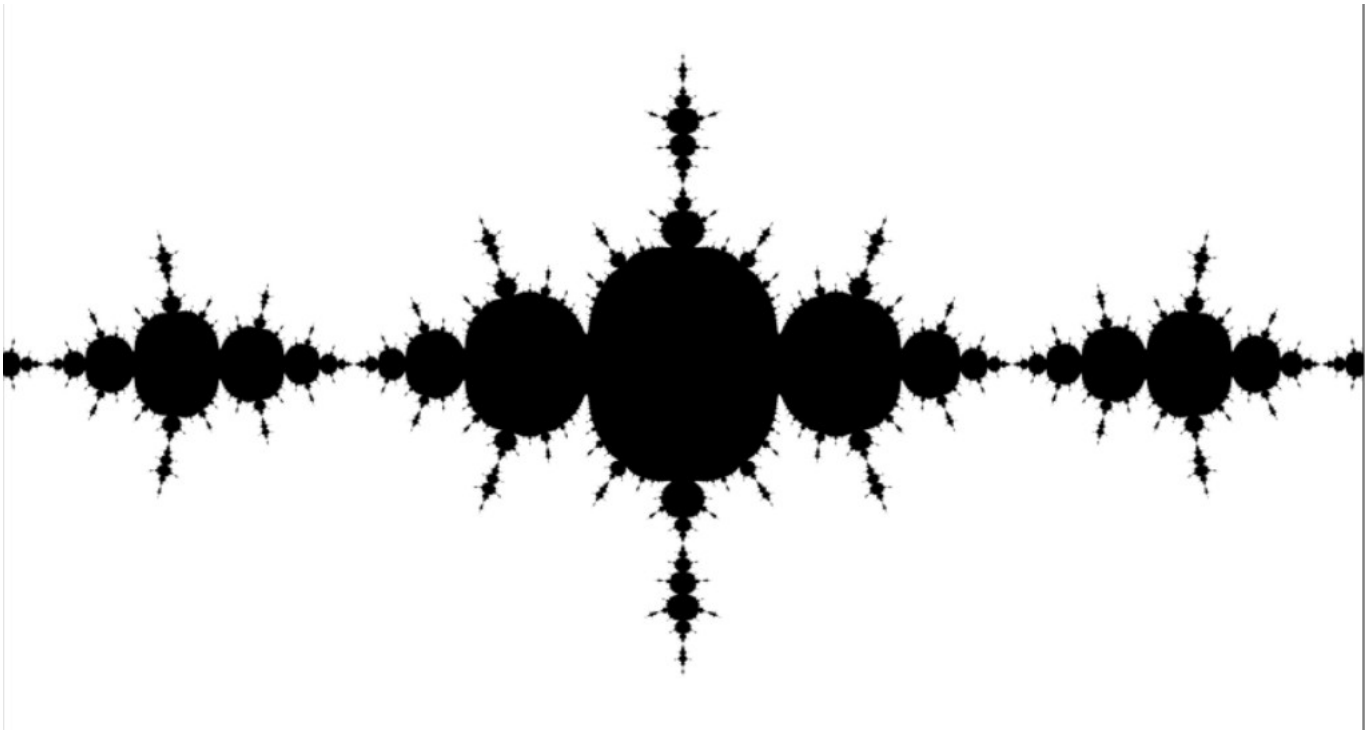

Рис. 3.12 Множина Жюліа для c = -1.25

На останок отримаємо зображення в повишеній якості, а саме щільність точок по дійсній та уявній частині дорівнюватиме 5000 при -1.5 ≤ [Real(c), Img(c)] ≤ 1.5. А також задамо кількість ітерацій рівною 1000 раз.

Рис. 3.13 twilight

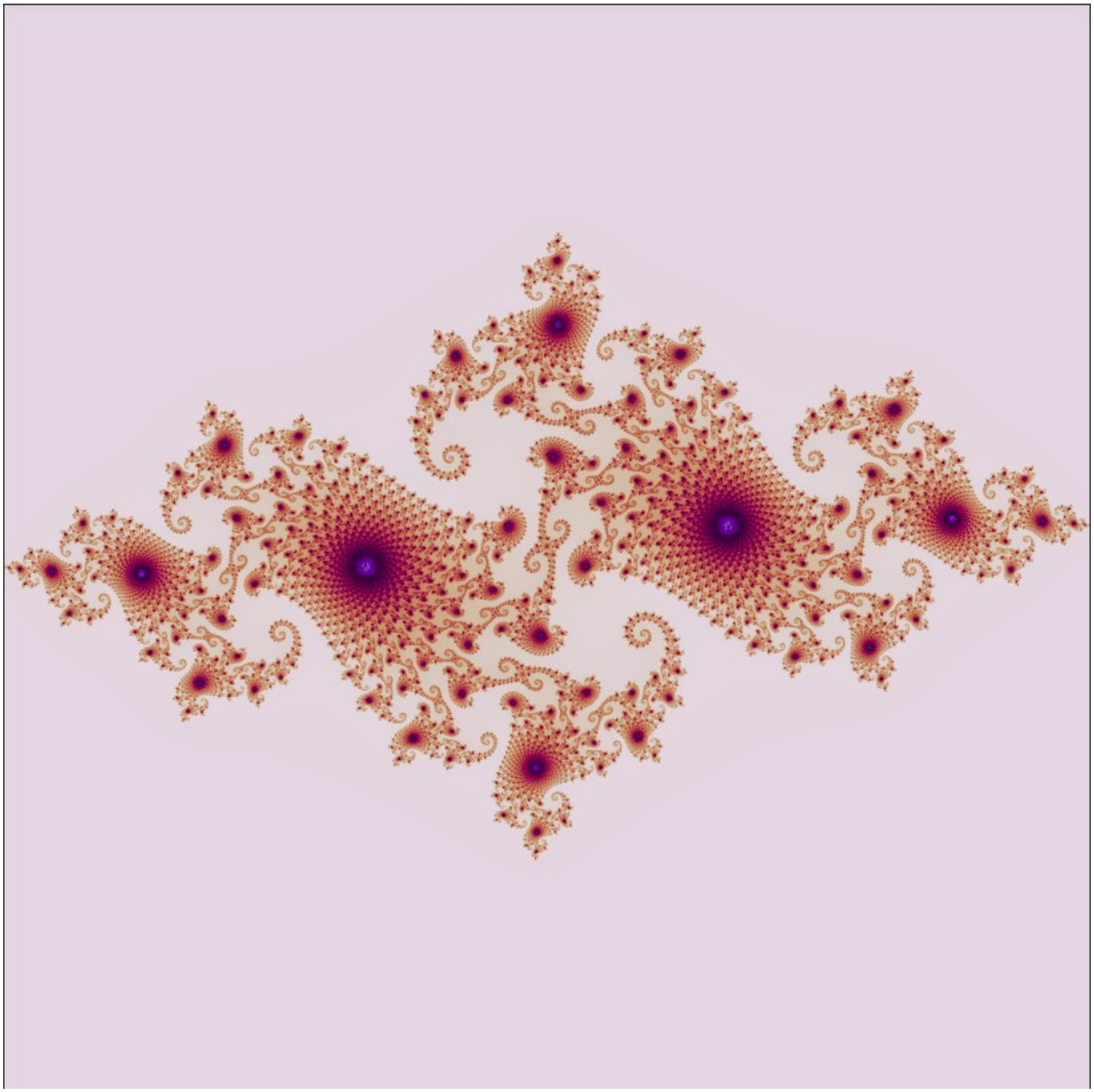

Рис. 3.14 Множина Жюліа для c = -0.74543 + 0.11301i

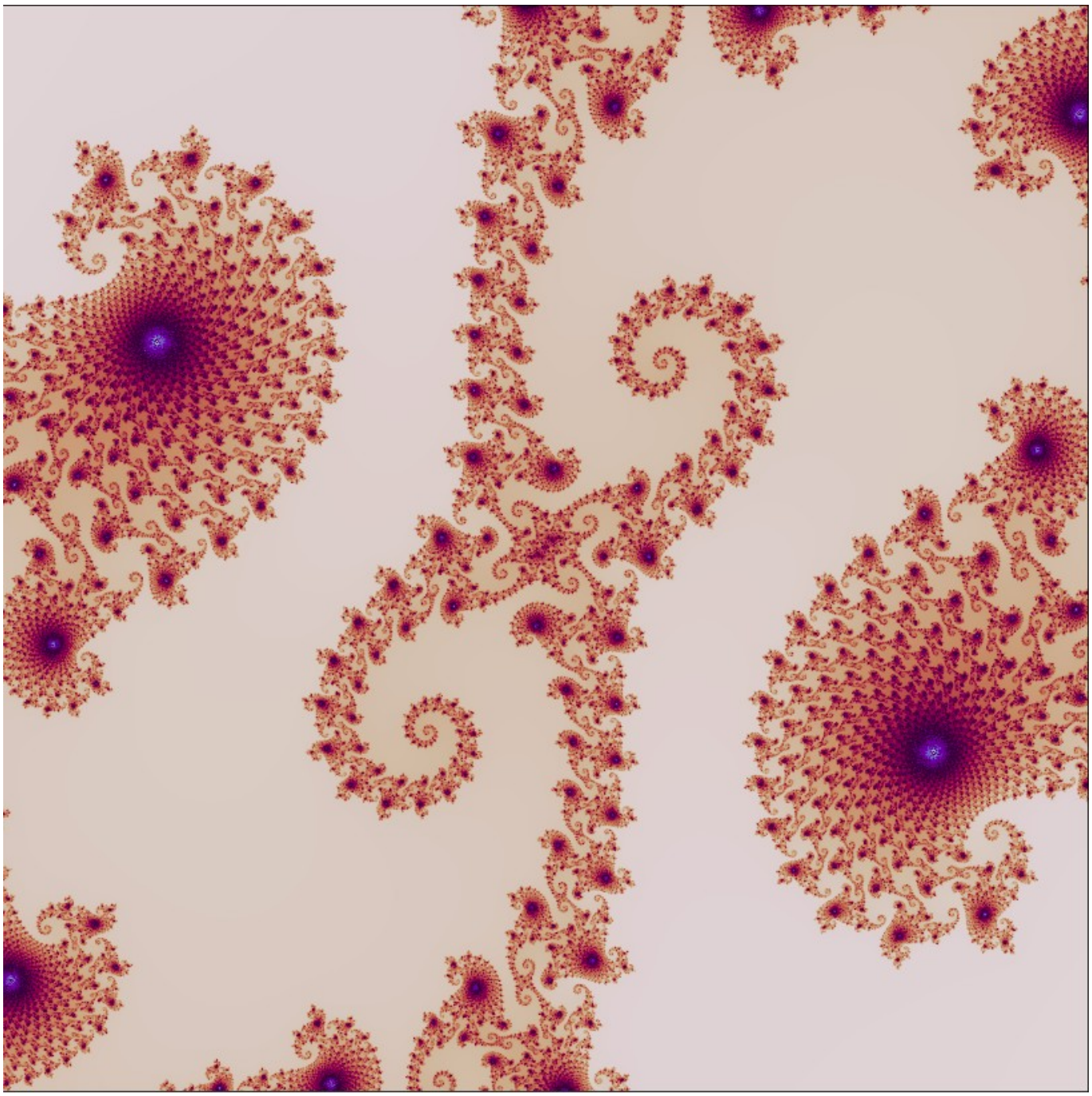

Рис. 3.15 Множина Жюліа для c = -0.74543 + 0.11301i в околі точки 0

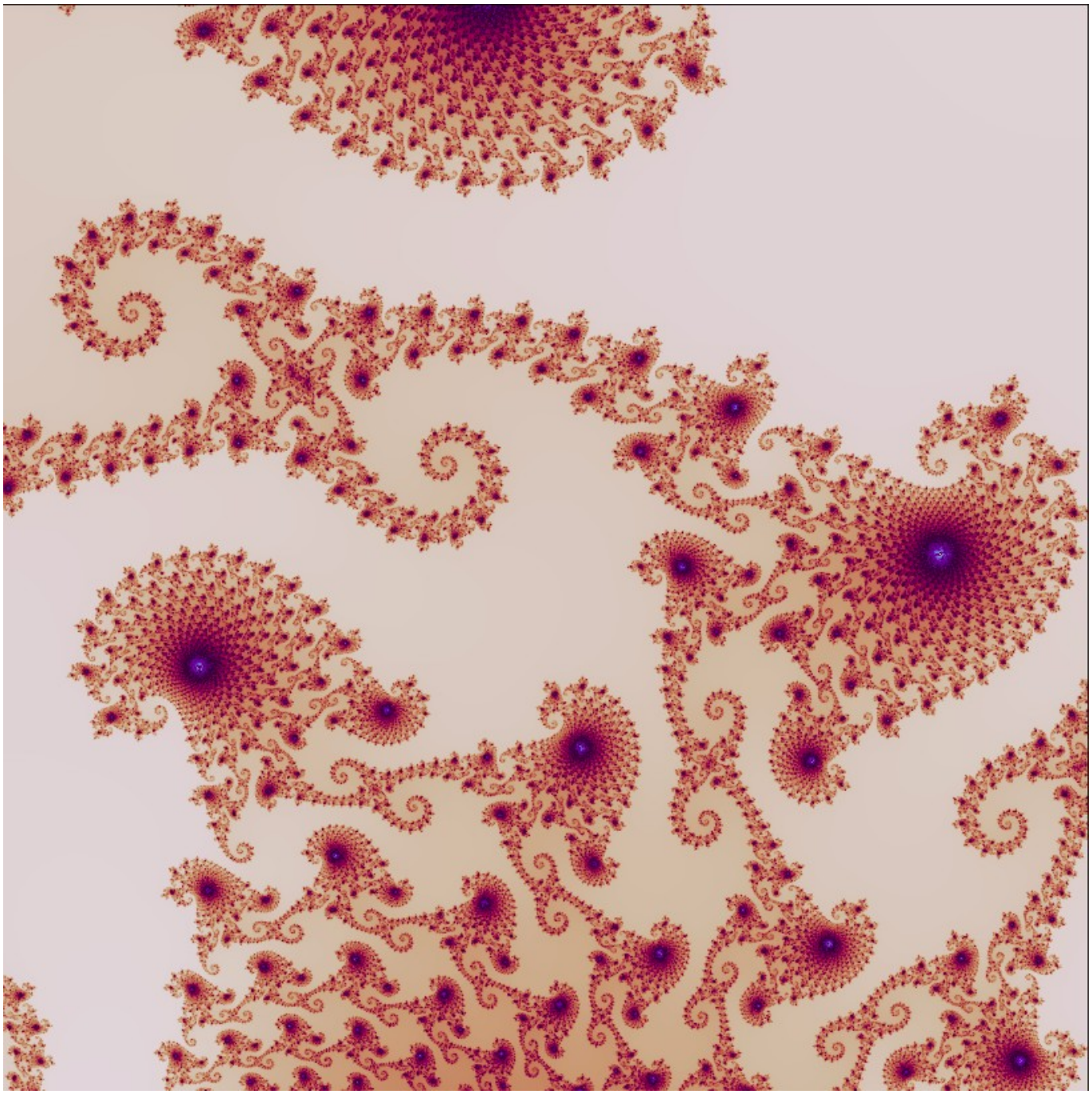

Рис. 3.16 Множина Жюліа для c = -0.74543 + 0.11301i в околі точки 0.4

ВИСНОВКИ

В межах даної лабораторної роботи було розроблено програмне забезпечення, яке дозволяє побудувати множини Жюліа, а також визначити характер нерухомих точок.

Варто звернути увагу, що порядок періодичної точки знайти не вдалося. Це пов’язано насамперед з програмними обмеженнями. Після проведення k-ітерації отримане значення від функції $f(z_{k-1})$ не співпадає з z_0 через накоплену похибку. Дану проблему, можна було вирішити символьним програмуванням, але навіть не зважаючи на тривалий пошук коренів композиції функцій, отриманий розв’язок важко інтерпретувати через громіздке аналітичне представлення.

Використовуючи колірні карти, можна отримати додаткову інформацію про характер нерухомих точок, напрям збіжності, а також захоплюючі ілюстрації.

Додаток А

```
import sympy
import numpy as np
import matplotlib.pyplot as plt


def inform(c):
    z = sympy.symbols('z', imaginary=True)

    eq = sympy.Eq(z, z ** 2 + c)
    sols = sympy.solveset(eq, z)

    for i, zi in enumerate(sols):
        print('Point No{}: {}'.format(i, zi))

        zmodule = abs(sympy.diff(z**2 + c, z).subs(z, zi))
        if zmodule < 1:
            type_point = 'притягуюча'
        elif zmodule > 1:
            type_point = 'відштовхуюча'
        else:
            type_point = 'нейтральна'

        print('Характер нерухомої точки: {}\n'.format(type_point))


def julia(c, cxpoint = 0, cypoint = 0, zoom = 1, xpoints = 2000,
          ypoints = 2000, max_iterations = 50, infinity_border = 4, use_k =
True):

    xmin = cxpoint - 1.5 * zoom
    xmax = cxpoint + 1.5 * zoom

    ymin = cypoint - 1.5 * zoom
    ymax = cypoint + 1.5 * zoom

    image = np.zeros((xpoints, ypoints))
    x, y = np.mgrid[xmin:xmax:(xpoints*1j), ymin:ymax:(ypoints*1j)]
```

```
    z = x + 1j * y
    for k in range(max_iterations):
        z = z**2 + c
        mask = (np.abs(z) > infinity_border) & (image == 0)

        image[mask] = k if use_k else 1
        z[mask] = np.nan

    return -image.T


#c = 0.31 + 1j * (0.4)
#c = -0.11 + 1j * (0.6557)
#c = -0.12 + 1j * (0.74)
#c = -0.194 + 1j * (0.6557)
c = -0.74543 + 1j * (0.11301)
#c = -1.25
#c = -0.481762 + 1j * (-0.531657)
#c = -0.39054 + 1j * (-0.58579)
#c = -0.15652 + 1j * (-1.03225)
#c = 0.11031 + 1j * (-0.67037)

fig, axes = plt.subplots(2, 2, figsize=(40,40))

inform(c)
image = julia(c,  max_iterations = 200, use_k = False)
axes[0, 0].imshow(image, cmap='Greys')

image = julia(c, max_iterations = 200)
axes[0, 1].imshow(image, cmap='afmhot')
axes[1, 0].imshow(image, cmap='cubehelix')
axes[1, 1].imshow(image, cmap='twilight')

for i in range(2):
    for j in range(2):
        axes[i, j].set_xticks([])
        axes[i, j].set_yticks([])


plt.figure(figsize=(15, 15))

inform(c)
```

```
image = julia(c, use_k = False)

plt.xticks([])
plt.yticks([])
plt.imshow(image, cmap='Greys')


plt.figure(figsize=(15, 15))

inform(c)
image = julia(c, max_iterations = 300)

plt.xticks([])
plt.yticks([])
plt.imshow(image, cmap='afmhot')
```