计算物理 作业报告2

PB14203209 张静宁

第二题

在复平面上任选一个参数 $C=a+ib$，画出该 C 值下的Julia集(图形可彩色，也可黑白或灰度)。

黑白Julia图

算法思路

复平面上任意一点 $Z=x+iy$，利用方程 $Z_{n+1}=Z_n^2+C$ 多次迭代不逃离的复数集合为 Julia 集，对应复平面上的点集，可能是美丽的图形。

迭代方程展开得到方程(1)，并拆开得到两个实数方程$(2)(3)$：

$$Z_{n+1}=(x_n^2-y_n^2)+a+i(2x_ny_n+b) \tag{1}$$

$$x_{n+1}=(x_n^2-y_n^2)+a \tag{2}$$

$$y_{n+1}=2x_ny_n+b \tag{3}$$

判断复平面上的点(x,y)是否属于 Julia 集的条件：

将(x,y)代入方程$(2)(3)$，迭代次数为Num，判断迭代后 $z^2=x^2+y^2$ 是否小于某个半径 R，当原点$(0,0)$是方程吸引子时，还应该把那些多次迭代趋于原点的点去掉。故判断条件为

$$ZERO<z_2<RADIUS \tag{4}$$

方程(4)中逃离半径在程序中设置为 $RADIUS=200$ ，$ZERO=0.00000001$

由于实际局限，并不能遍历整个复平面，故程序中计算一个正方形区域 $x\in[-2,2],y\in[-2,2]$ 上的Julia集

同样，必须设定x或y相隔多大的步长取一个点，程序中将步长 $STEP=0.001$

复平面上Julia集的形状取决于复常数C，我们称为复数C的Julia集。另外迭代次数 Num 会影响到得到的点的多少，迭代越多次得到的点数越少。

程序使用说明

文件说明

编程环境：Ubuntu，要预装 C 和 Python

- julia.c 为用 C 语言编写的计算程序，负责计算并输出数据点
- julia 为 C 语言编译后的可执行文件
- plot.py 为用 Python 编写绘图脚本

具体操作

分别执行以下三条命令：

```
$ gcc julia.c -o julia          # 编译 julia.c 程序
$ ./julia > data                # 在当前目录下执行 julia 程序，并将结果输出到 data 文件
$ python plot.py                # 执行绘图程序，将生成的图片保存为pdf和png格式，并复制数据文件
```

在执行第二条命令后会输出如下信息，提醒用户输入相关参数：

```
This program calculates the Julia Set of C = a + i b.
Input a between [-1,1]: -1                          # 输入 a 的值
Input b between [-1,1]: 0                           # 输入 b 的值
Input number of iteration for each points: 30       # 输入迭代次数 Num 的值
Totally 5396 Points!                                # 本次一共输出 5396 个数据点
```

操作时，需要多次尝试调整Num的大小，一般来说取30左右，使得输出的点数为1万~10万比较合适。

在执行第三条命令后，可以在results文件夹中得到三个文件：

- *.txt 当前条件下得到的数据文件
- *.pdf 当前条件下得到的图片
- *.png 当前条件下得到的图片

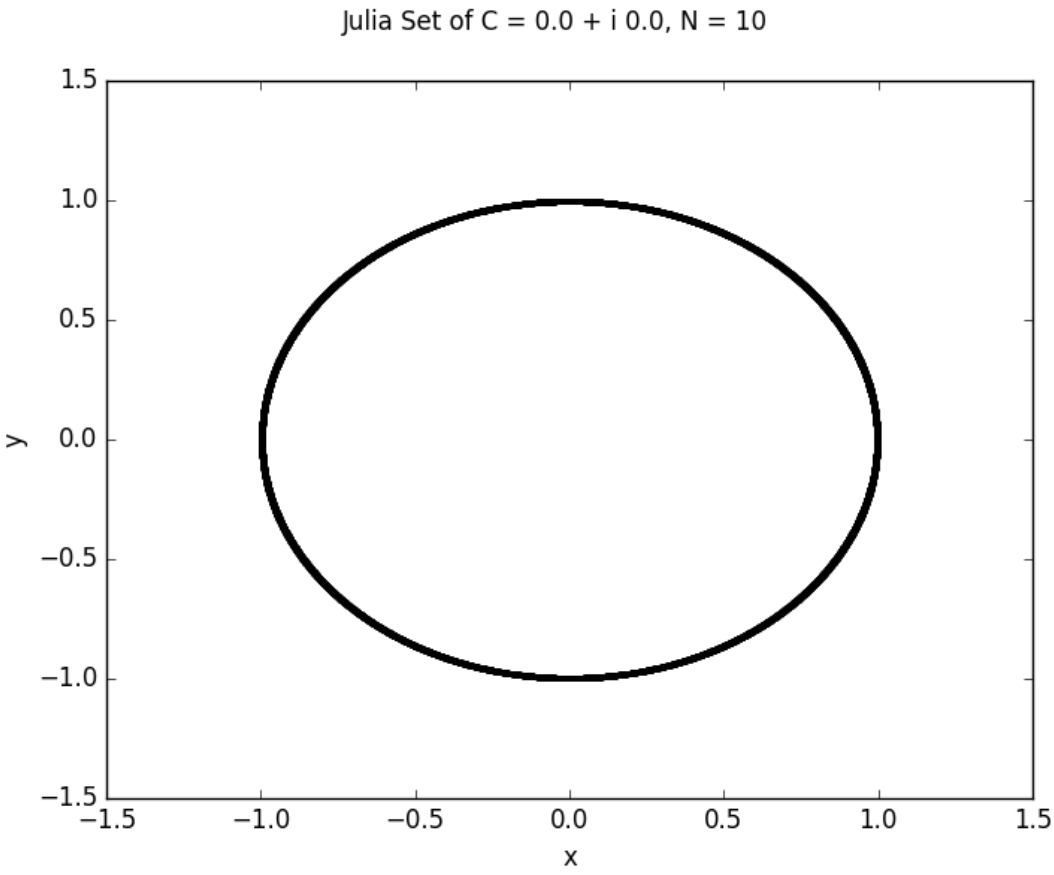
文件名*为输入参数的组合，可以由文件名看出输入参数值。

说明：在做图的具体过程中，多次改变输入参数，改变Python绘图脚本可以得到不同效果的图片。以上做图过程中 plot.py 曾被修改，故和目前提交提交的版本可能有区别。

计算结果与分析

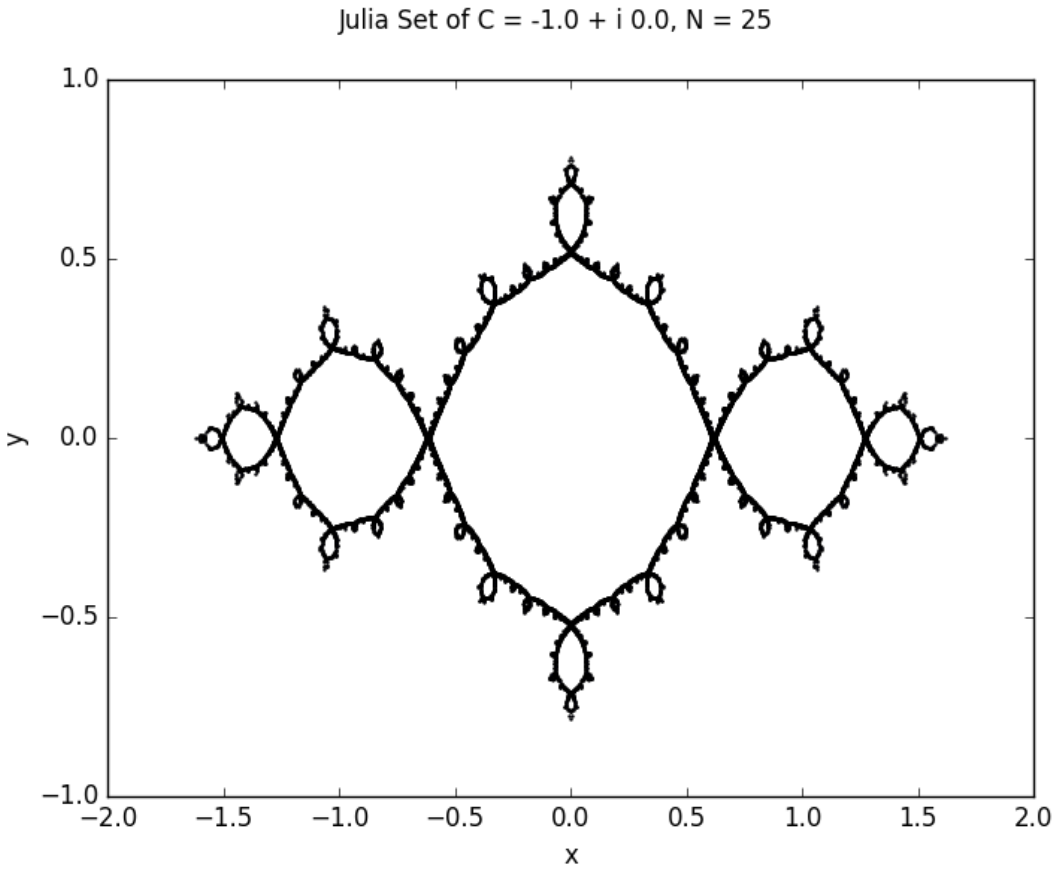
C = 0

$C = 0$ 时，原点和无穷远点是吸引子，需要扣除趋于原点的点。

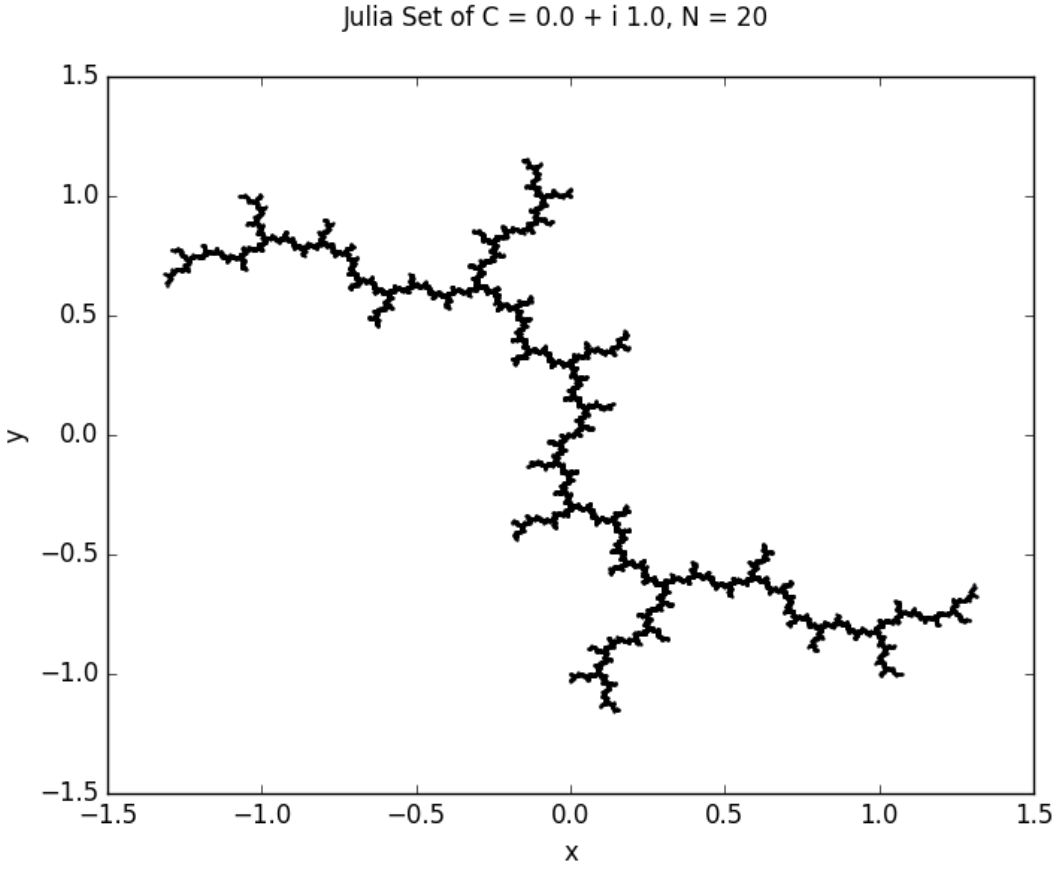


C = -1

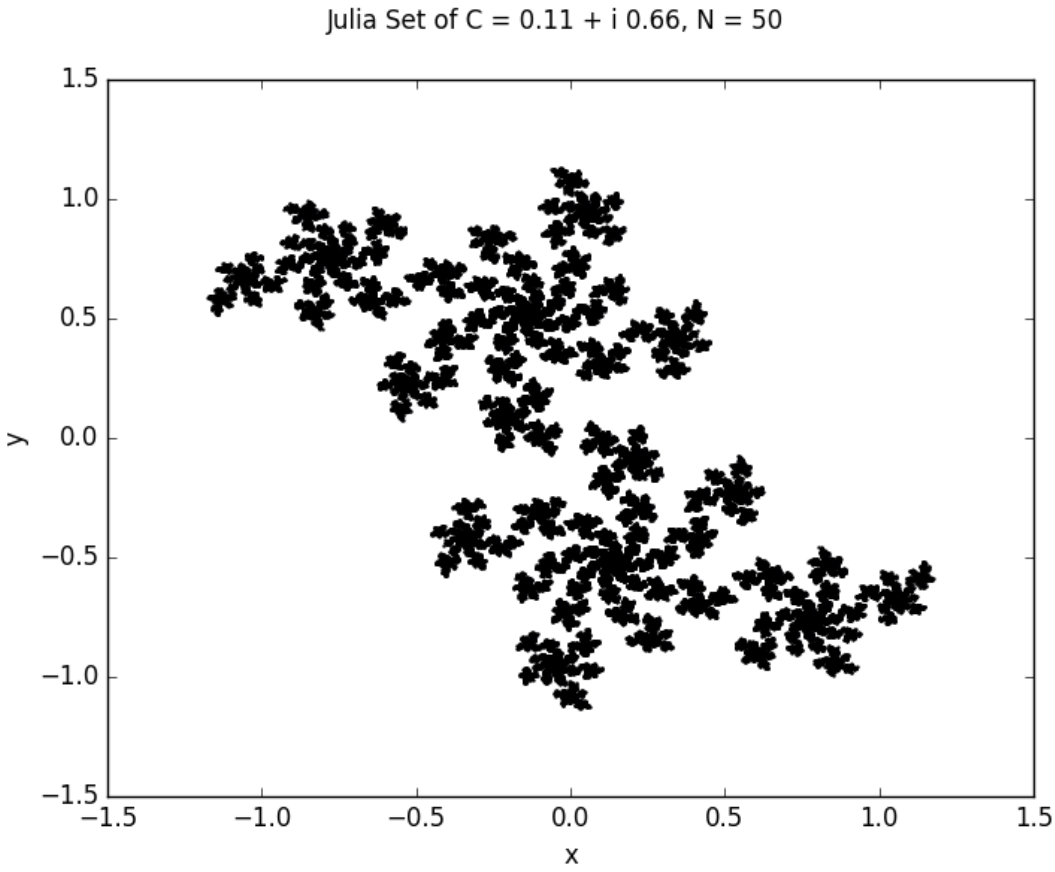
$C = -1$时，原点也是吸引子


Julia Set of C = -1.0 + i 0.0, N = 25
1.0
0.5
0.0
−0.5
−1.0
y
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
x

C = i

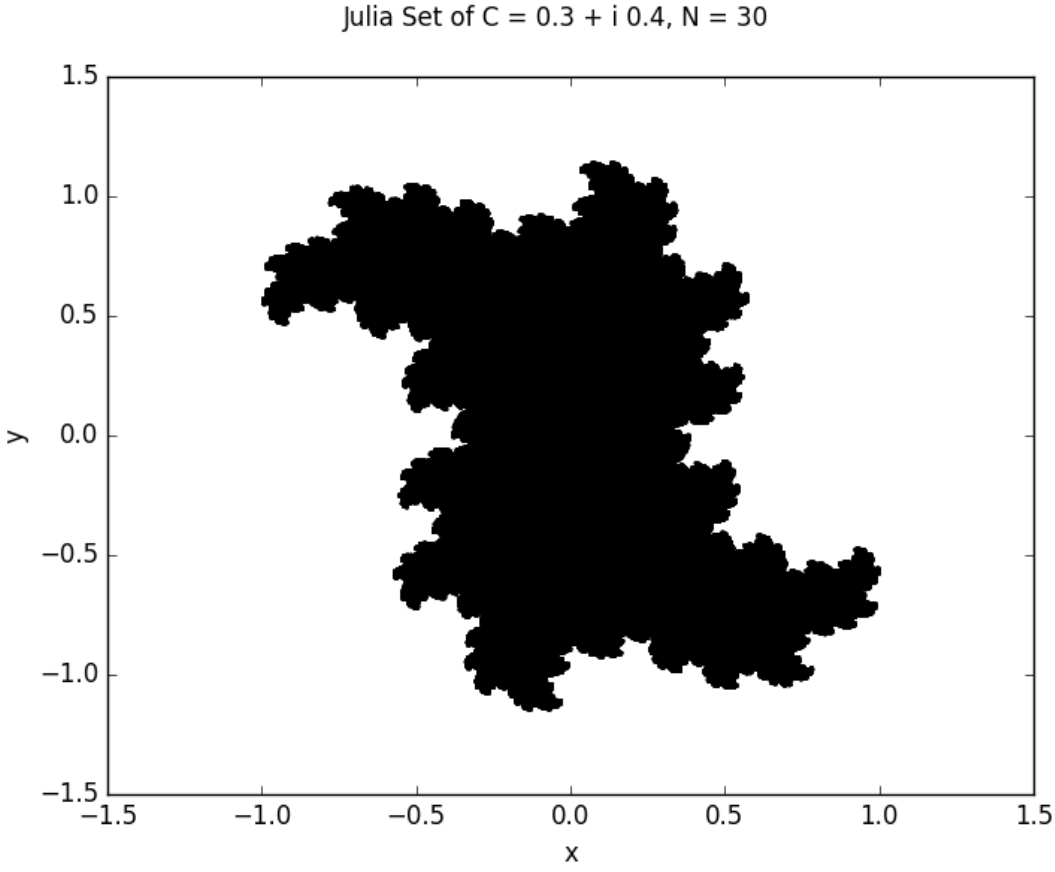

Julia Set of C = 0.0 + i 1.0, N = 20
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
y
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
x

C = 0.11 + i 0.66



C = 0.3 + i 0.4

说明此时有除原点以外的吸引子，手动将程序中 $ZERO = 0.00000001$ 改成 $ZERO = 0.1$ 可以把内部的点挖掉一些。见后面彩色Julia图。



彩色Julia图

简要说明

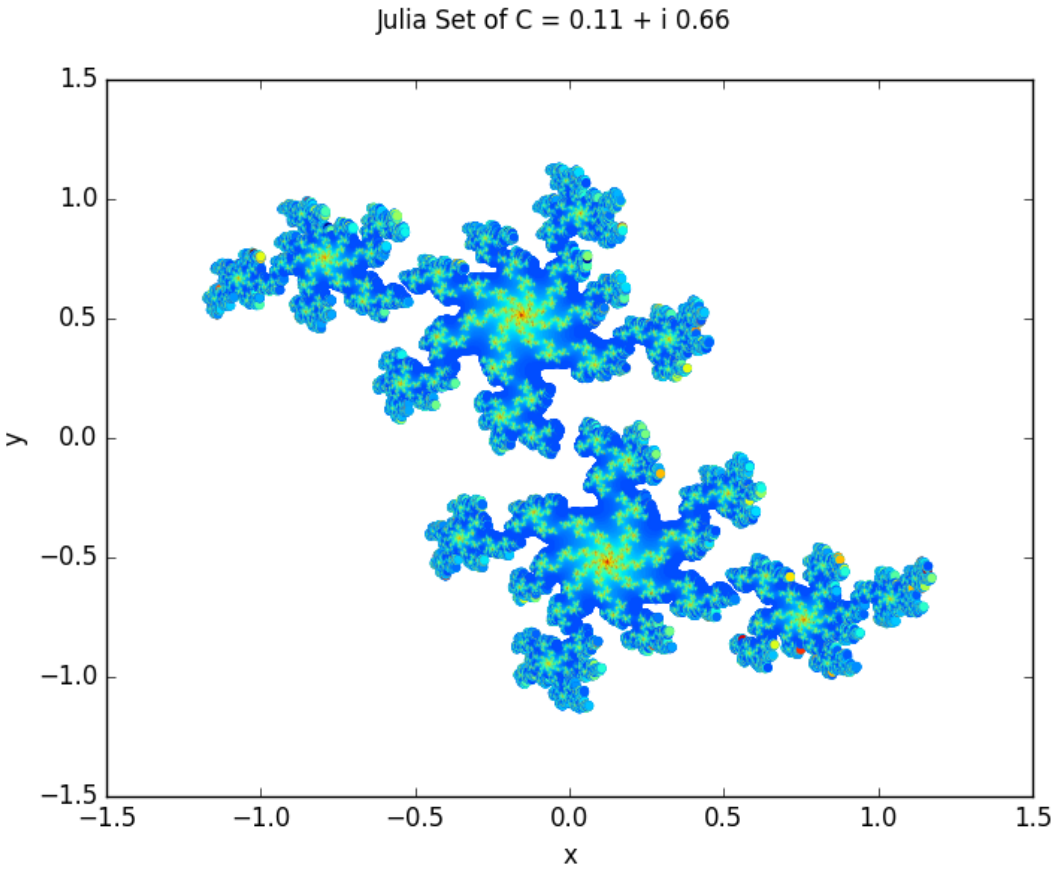
彩色Julia图的算法思路，程序操作都与黑白Julia图类似。唯一不同的在于彩色Julia图，需要知道每个点逃逸的迭代次数N。并根据逃逸速度的大小（N来衡量），绘上不同的颜色。实现上面，C语言程序julia.c需要修改，绘图脚本plot.py也需要做相应的修改。以下只做简要说明。

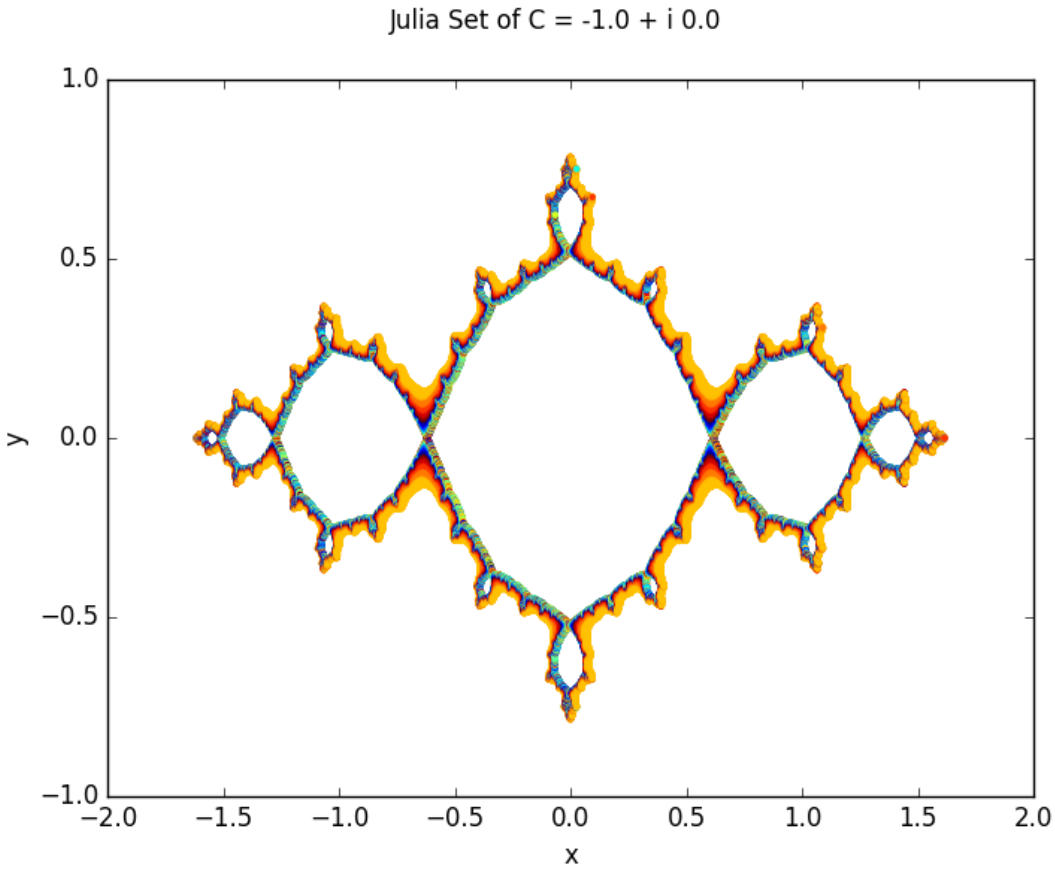
- color.c 为修改后的C语言计算程序
- color_plot.py 为修改后python绘图脚本

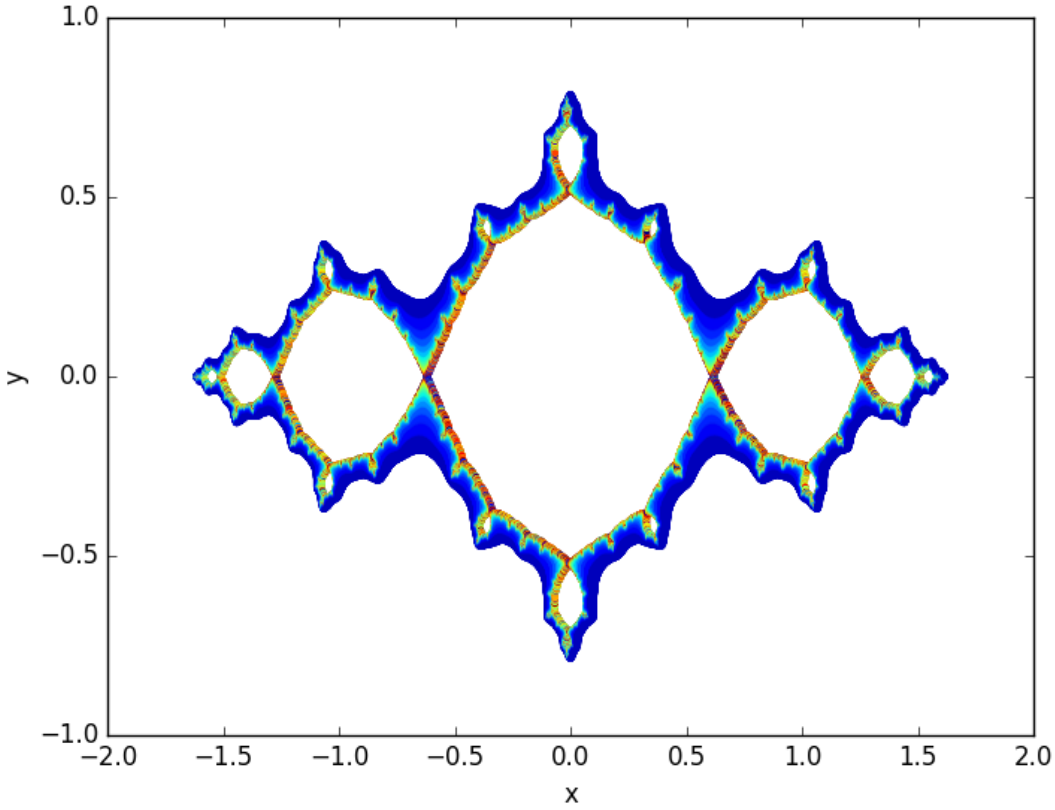
为了让彩色图片更美观，根据每个点迭代次数N赋值的颜色应该是渐变的，即N的值越相近，颜色应该越相似。

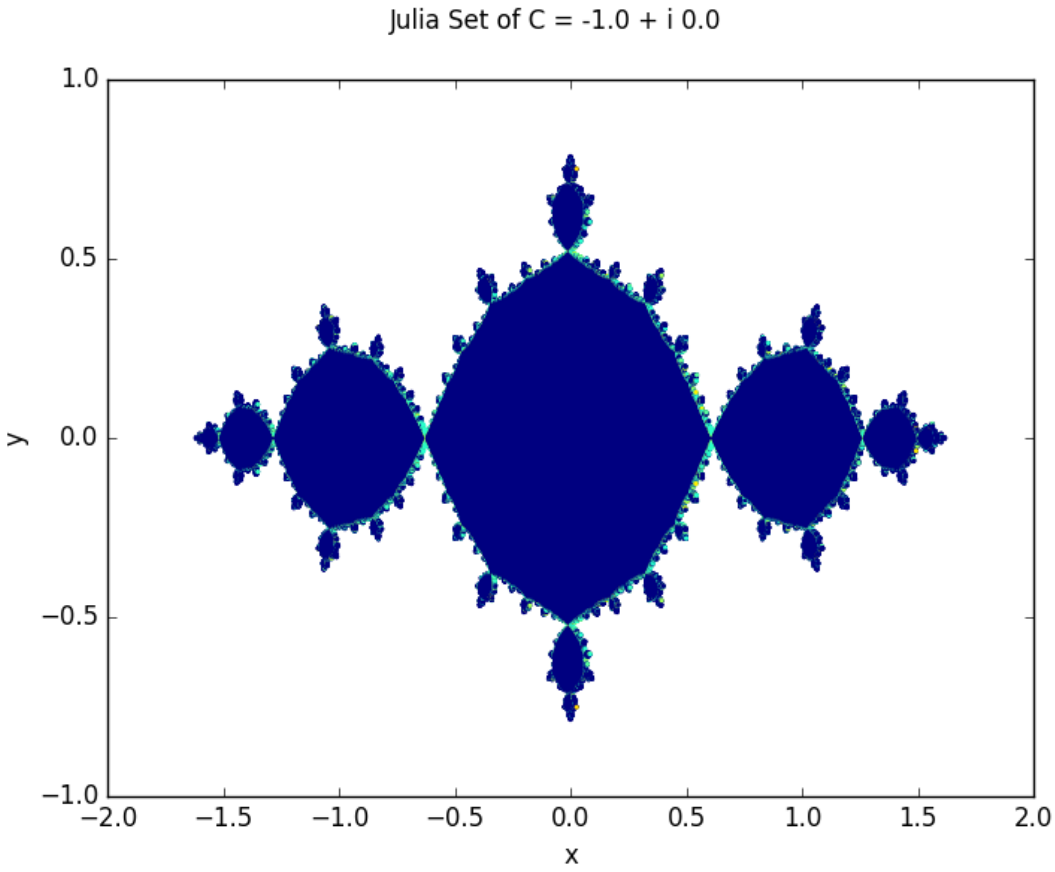
计算结果

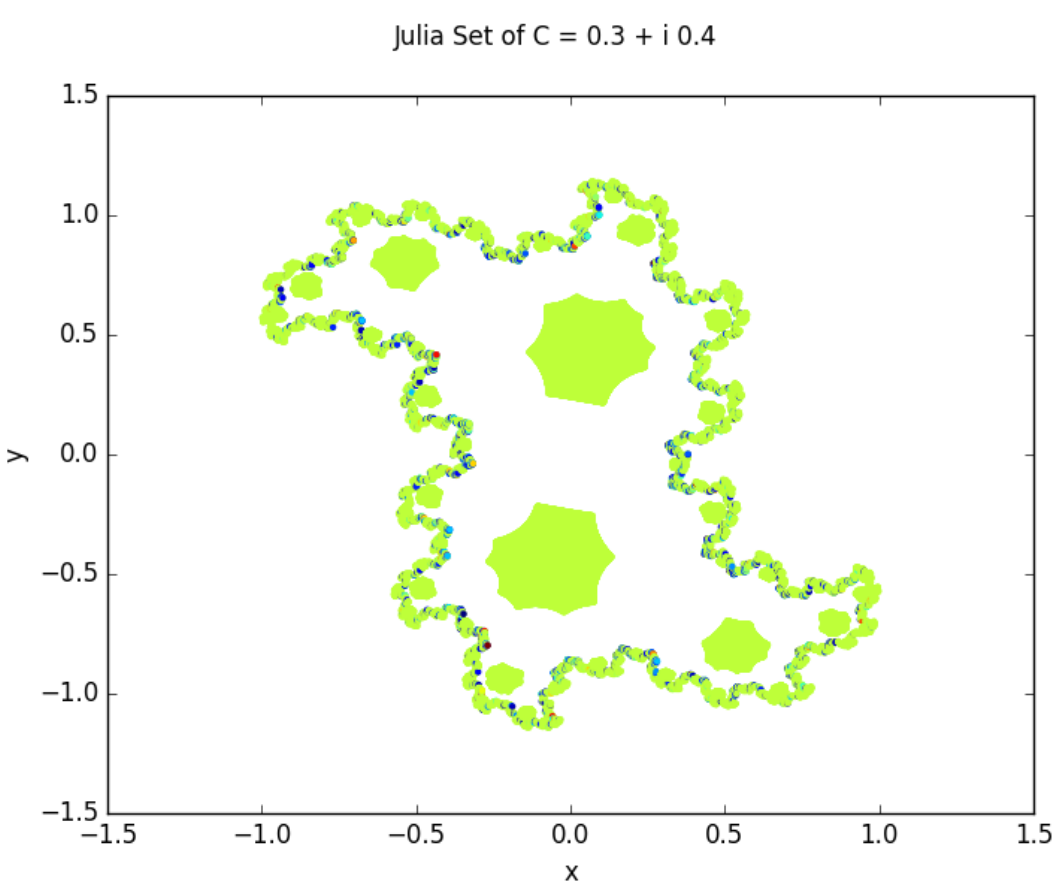
需要根据每一个Julia集的图形特点，单独修改python绘图脚本中的颜色参数，甚至需要修改color.c里的$ZERO$。总之为了使图片更好看，要多尝试。

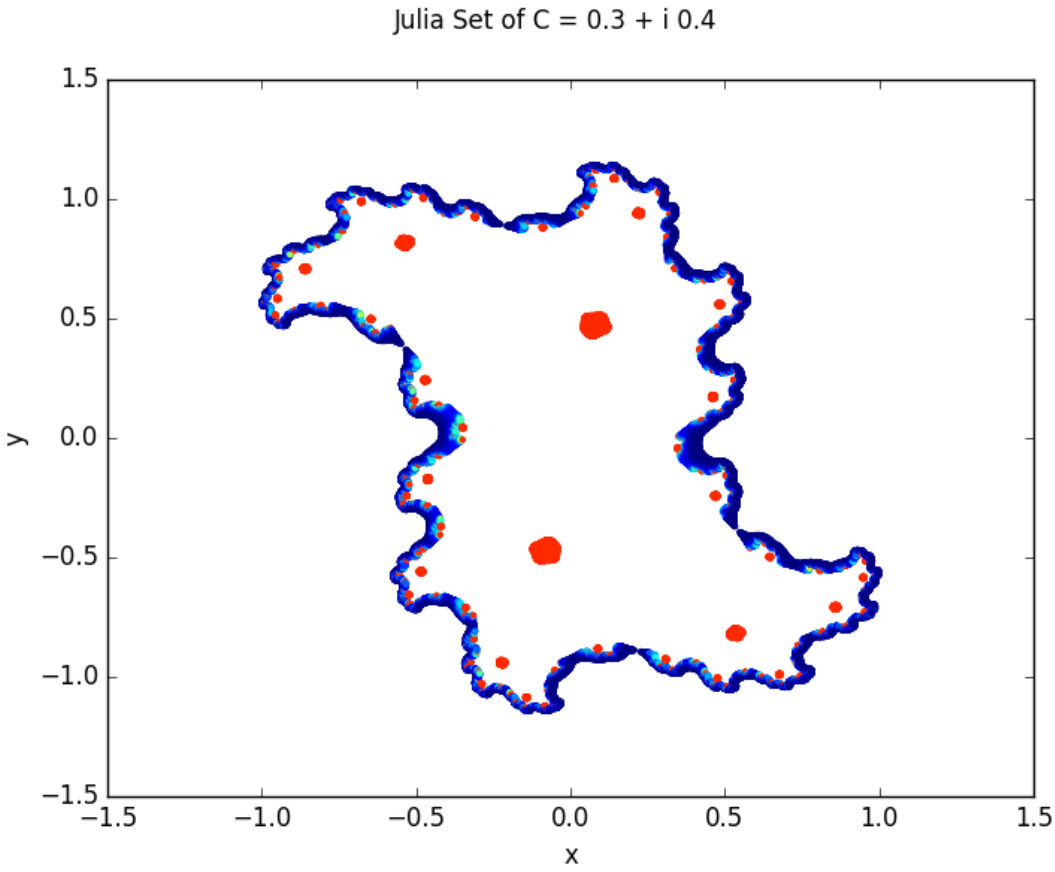



Julia Set of C = -1.0 + i 0.0
1.0
0.5
0.0
−0.5
−1.0
y
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
x


Julia Set of C = -1.0 + i 0.0
1.0
0.5
0.0
−0.5
−1.0
y
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
x


Julia Set of C = -1.0 + i 0.0
1.0
0.5
0.0
−0.5
−1.0
y
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
x


Julia Set of C = 0.3 + i 0.4
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
y
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
x


Julia Set of C = 0.3 + i 0.4
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
y
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
x