02：Julia

许传奇 PB16021546

1 题目

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）。

2 原理与算法

2.1 原理

2.1.1 迭代法

如果一个物理量的表达式中含有该物理量本身：

$$x = f(x) \tag{1}$$

求解这个物理量时，通常采用数学上的迭代法。

当变量为多元变量时，同样可以使用迭代法。

例如：

$$\begin{cases} x_{n+1} = f(x_n, y_n) \\ y_{n+1} = g(x_n, y_n) \end{cases} \tag{2}$$

也可以运用迭代法求解。

2.1.2 Julia集

Julia集的迭代方程为：

$$Z_{n+1} = Z_n^2 + C \tag{3}$$

其中C为复平面上给定的一个参数：

$$C = x_c + iy_c \tag{4}$$

平面上的任意一点$Z_0 = x + iy$，若使用迭代方程无限次，Z的模可能趋于无穷，或趋于0，或既不趋于无穷又不趋于0。

当这个点迭代无穷次后既不趋于无穷又不趋于0，这个点属于Julia集，即所有这样的点构成Julia集。

2.2 算法

1. 求解迭代方程：

   设$Z_n = x_n + iy_n, C = x_c + iy_c$，将其带入到迭代方程(3)中，得到：

   $$Z_{n+1} = (x_n^2 - y_n^2) + x_c + i(2x_ny_n + y_c) \tag{5}$$

   即得到Z_{n+1}的实部x_{n+1}与虚部y_{n+1}：

   $$\begin{cases} x_{n+1} = (x_n^2 - y_n^2) + x_c \\ y_{n+1} = 2x_ny_n + y_c \end{cases} \tag{6}$$

2. 计算迭代次数：

   取复平面上的一个正方形区域，其x、y的范围均是$(-range, range)$，设置步长step，运用循环即可遍历该平面上的点。同时我们设置num作为最大迭代次数，small_radius和large_radius作为判断的最小距离和最大距离。

   对循环中的每一点，我们以它的横、纵坐标作为x_0和y_0，代入到迭代方程中，不停迭代。

   直到满足以下两种情况中的一种:

   (a)
   $$Z_n^2 = x_n^02 + y_n^2 > large\_radius \tag{7}$$
   或
   $$Z_n^2 = x_n^2 + y_n^2 < small\_radius \tag{8}$$

   (b) 迭代达到num但仍未满足上述条件，即仍然满足：

   $$small\_radius < Z_n^2 = x^2 + y^2 < large\_radius \tag{9}$$

   时退出迭代方程的循环。

   当退出循环时迭代次数为num时，我们可以认为这个点属于Julia集。

   记下每个点退出循环时迭代的次数。将每个点的横、纵坐标、迭代次数输入到文件data.txt中。

3. 画图：

   使用blackplot.py、colorplot.py分别画黑白Julia集图和彩色Julia集图。

   其中黑白Julia集图的算法是读取data.txt中的数据，判断迭代次数是否为给定的迭代次数(即num)。若是，则认为该点在Julia集中；否则，则认为该点不在Julia集中。画出所有迭代次数为num的点的散点图，即为黑白Julia集图。

   彩色Julia集图的算法是读取data.txt中的数据，将画散点图的函数scatter的参数c(颜色参数)设置成迭代次数的数组，即不同的迭代次数代表不同的颜色。画出所有点的散点图，即为彩色Julia集图。

3 源文件使用说明

打开02Julia.exe执行文件，输入范围、步长、C的实部、C的虚部、逃逸距离和最大迭代次数，随后等待计算机计算。

计算完毕后，将得到data.txt文件，其中保存有每个点的横、纵坐标以及迭代的次数。其中第一行三个数据分别为范围、步长和最大的迭代次数。第二行的三个数据分别为C的实部、虚部和逃逸距离。从第三行开始，第一列是点的实部，第二列是点的虚部，第三列的迭代次数。

打开blackplot.py、colorplot.py，编译并运行，即可得到画黑白Julia集图和彩色Julia集图。

4 计算结果及具体分析

4.1 黑白Julia集图

分别设置不同的参数，绘制出相应的黑白Julia集图，如下面所示：

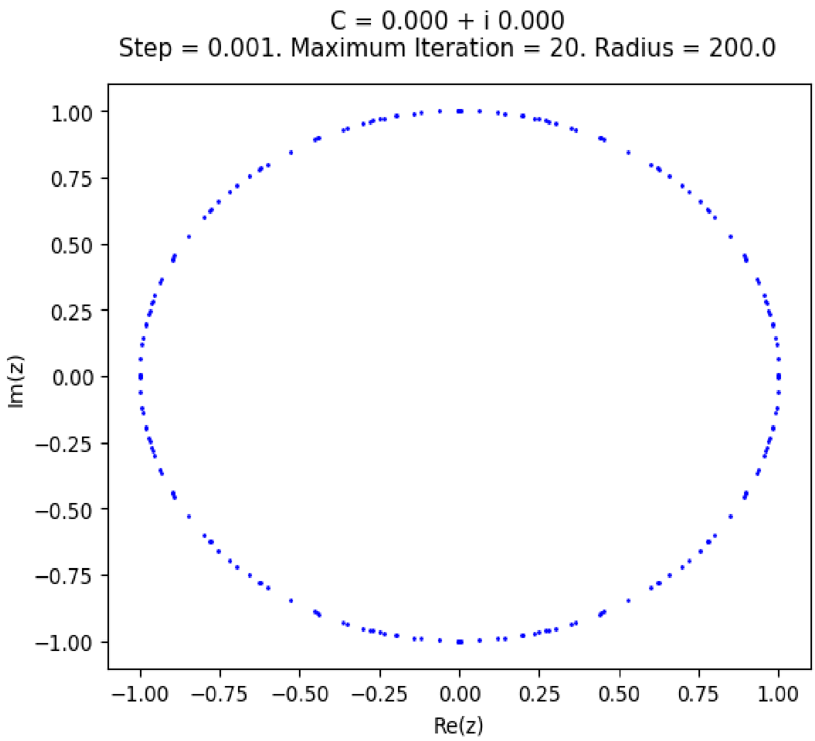

图 1: 黑白Julia图：$C = 0.000 + i0.000, num = 20$

对于$C = 0$的时候，Julia集应该是复平面上的一个单位圆，可是程序模拟的非常不好，圆上很多的点程序都认为不在Julia集中。

原因是我们选取的点是从x、y的最小处逐步加上步长step，可是圆上的大部分点不满足这样的取法。因此迭代次数较大时，哪怕这些点与圆上的点的距离非常小，经过迭代后，距离要么小于$small_radius$了，要么大于$large_radius$了，程序会提前退出迭代的循环。因此他们的迭代次数达不到num，画图的程序也就认为他们不在Julia集中，于是也就不能在图片上表示出来。

解决这个的办法是减小最大迭代次数num，我们可以得到更加"不精确"的Julia集。但因为图中的点与真正属于Julia集中的点相差很小，因此我们可以近似用这些图片来表示Julia集的点。如下图所示：

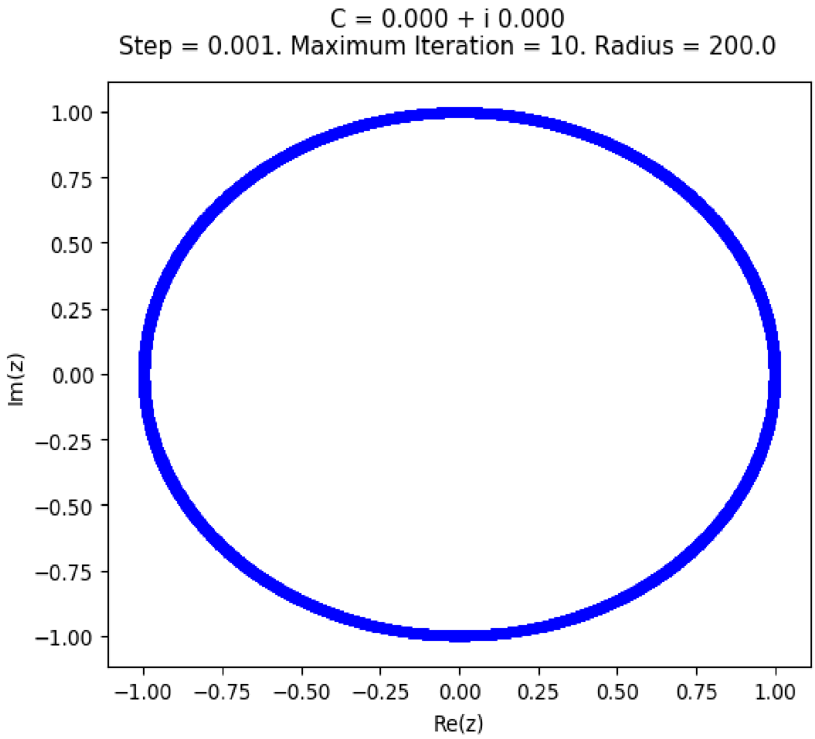

图 2: 黑白Julia图：$C = 0.000, num = 10$

这次图形更接近一个圆，说明我们这种减小判断准确度但提高图片的相似性的办法是可行的。

下面是一些其他C取值的图片：

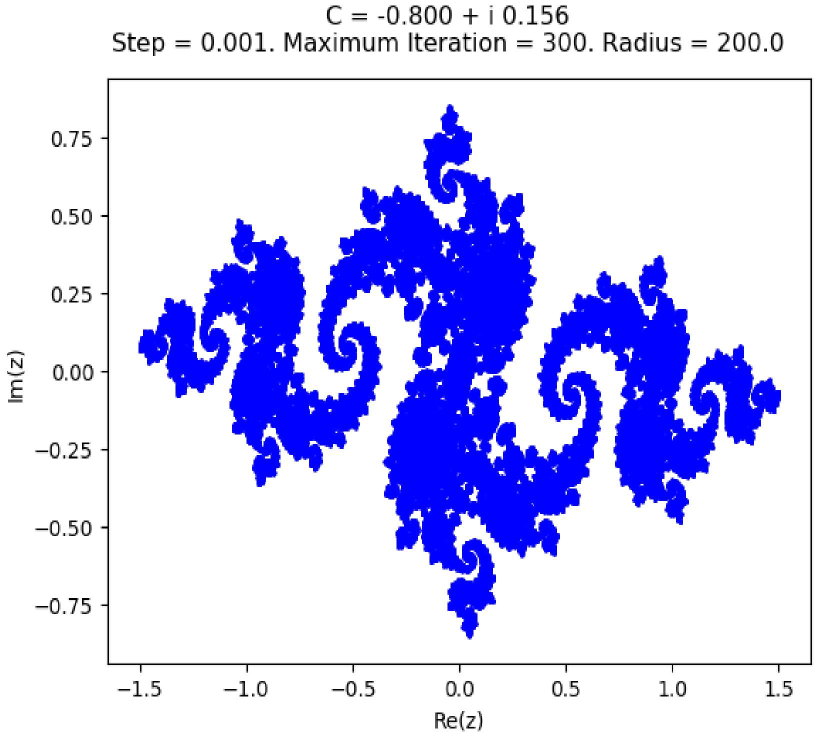

图 3: 黑白Julia图：$C = -0.080 + 0.156i, num = 300$

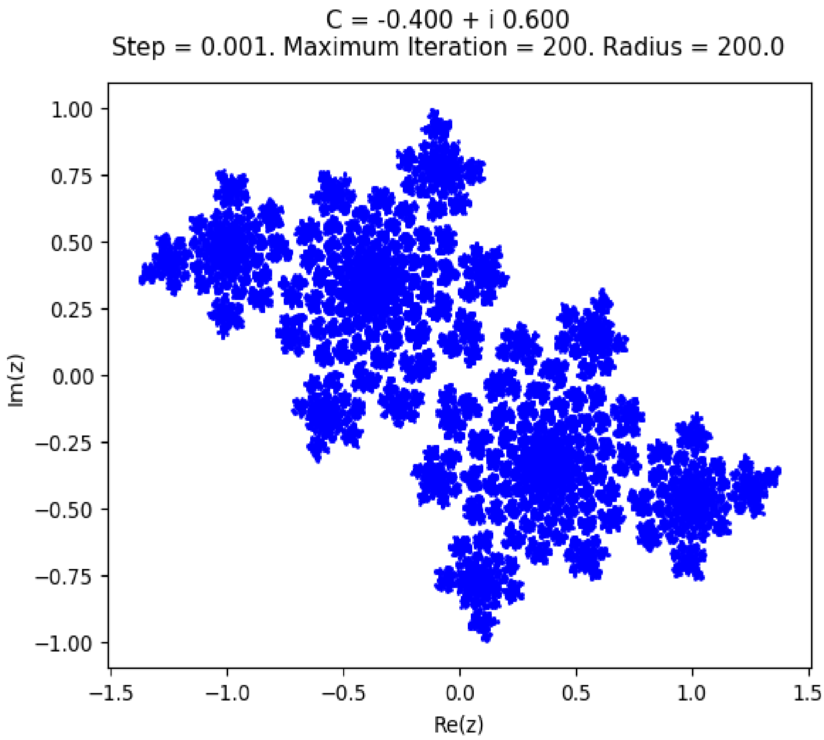

图 4: 黑白Julia图：$C = -0.400 + 0.600i, num = 200$

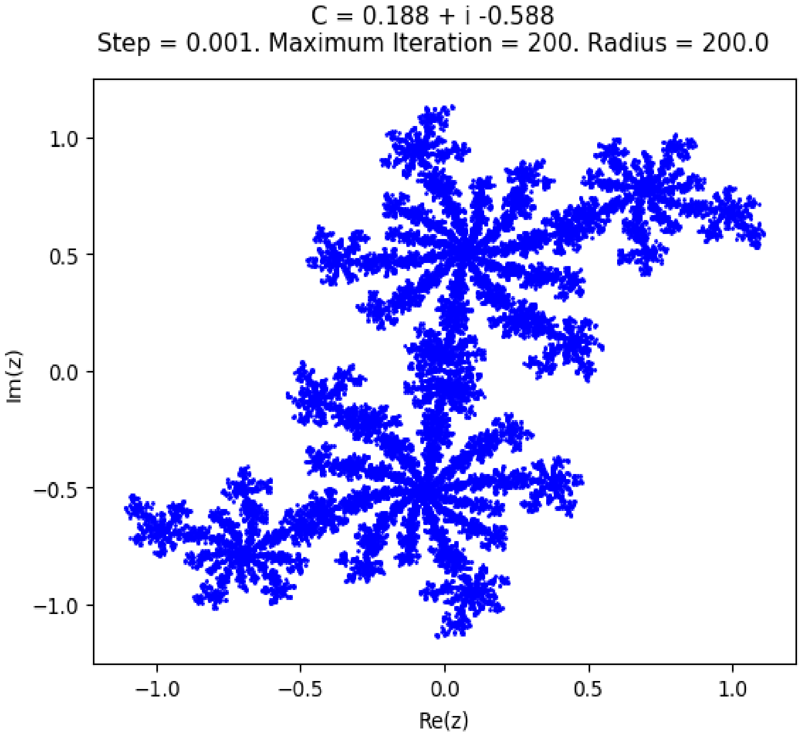

图 5: 黑白Julia图：$C = 0.188 - 0.588i, num = 200$

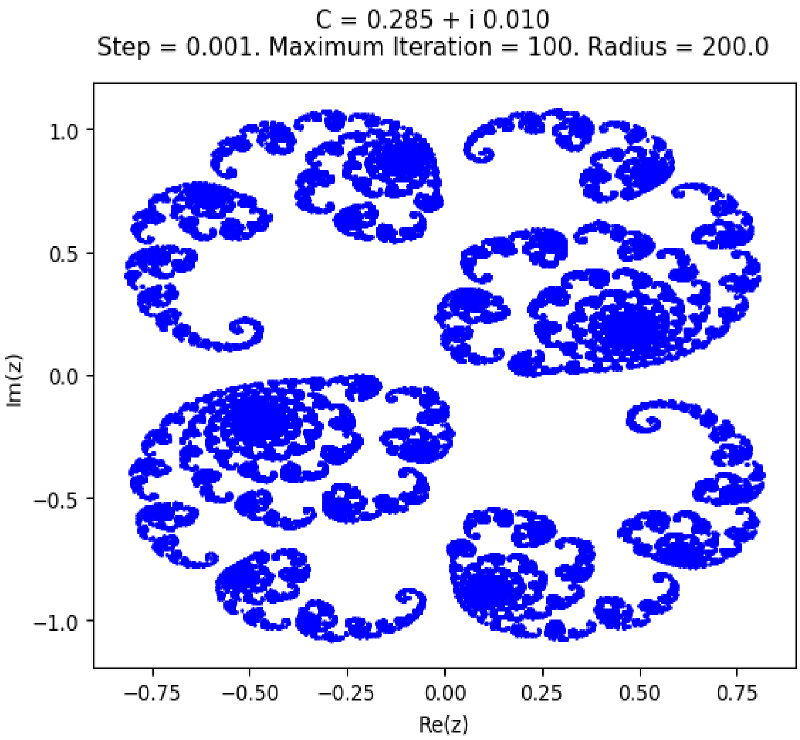

图 6: 黑白Julia图：$C = 0.285 + 0.010i, num = 100$

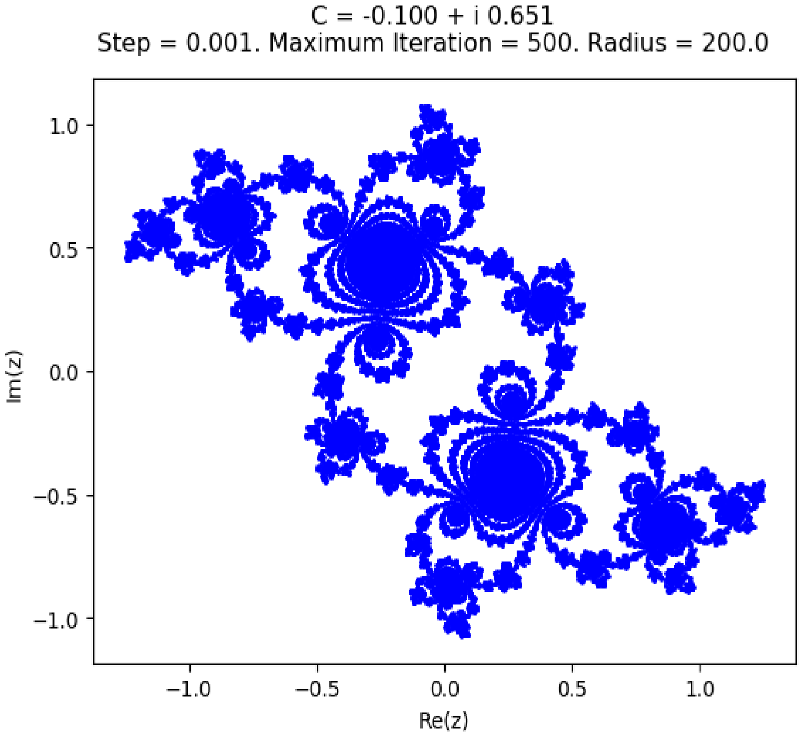

图 7: 黑白Julia图：$C = -0.100 + 0.651i, num = 500$

4.2 彩色Julia集图

我们不必像绘制黑白Julia集那样判断该点是否属于Julia集，而直接将迭代次数当作颜色数组的参数输入，即可直接画出彩色Julia集。

分别设置不同的参数，绘制出相应的彩色Julia集图，如下面所示：

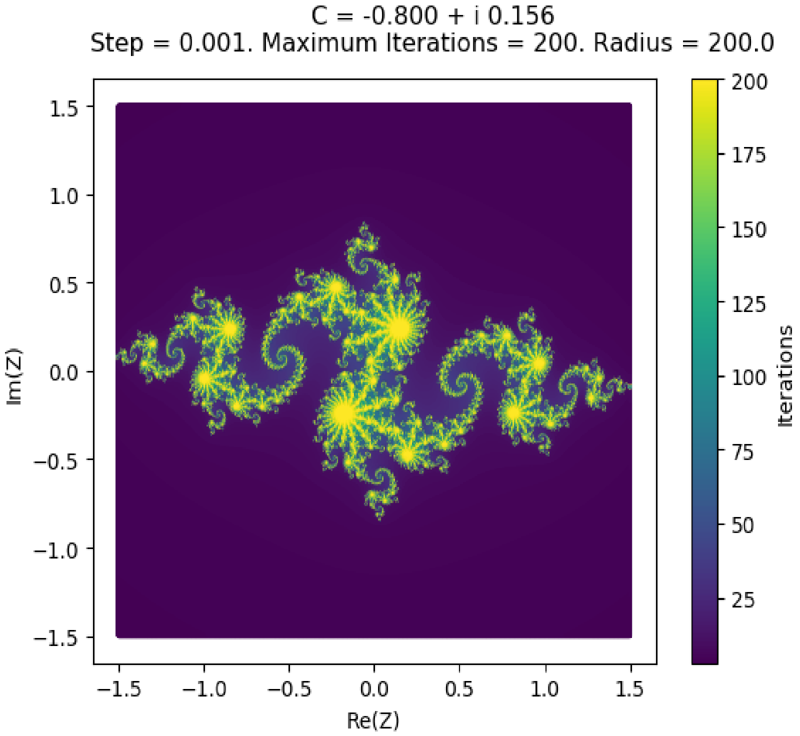

图 8: 彩色Julia图：$C = -0.800 + 0.156i, num = 200$

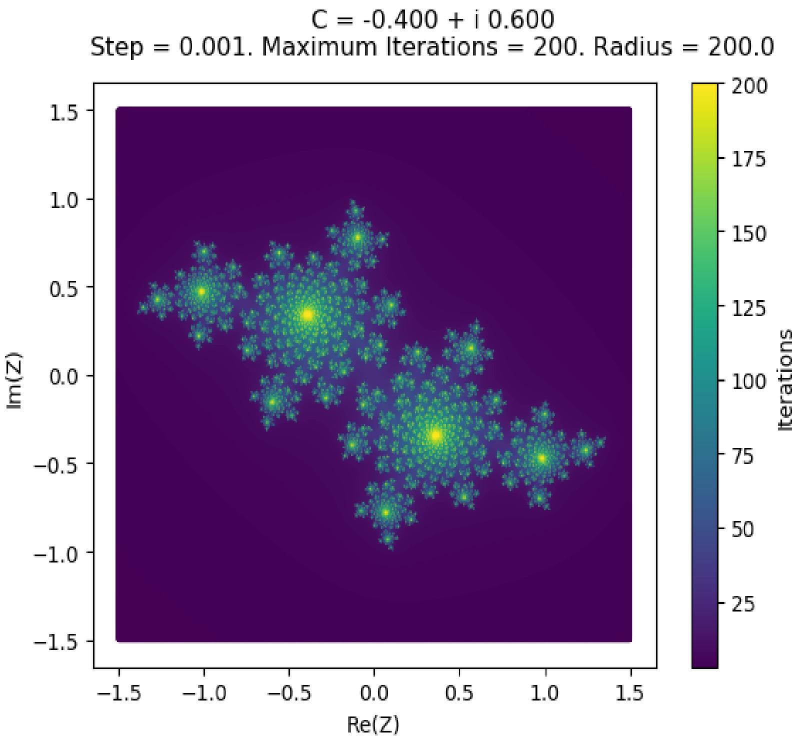

图 9: 彩色Julia图：$C = -0.400 + 0.600i, num = 200$

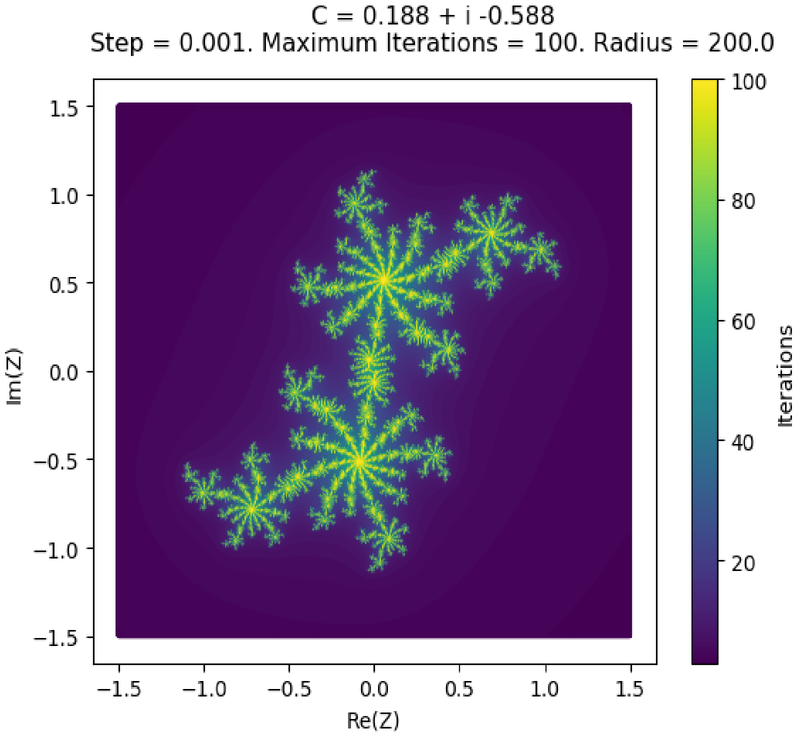

图 10: 彩色Julia图：$C = 0.188 - 0.588i, num = 100$

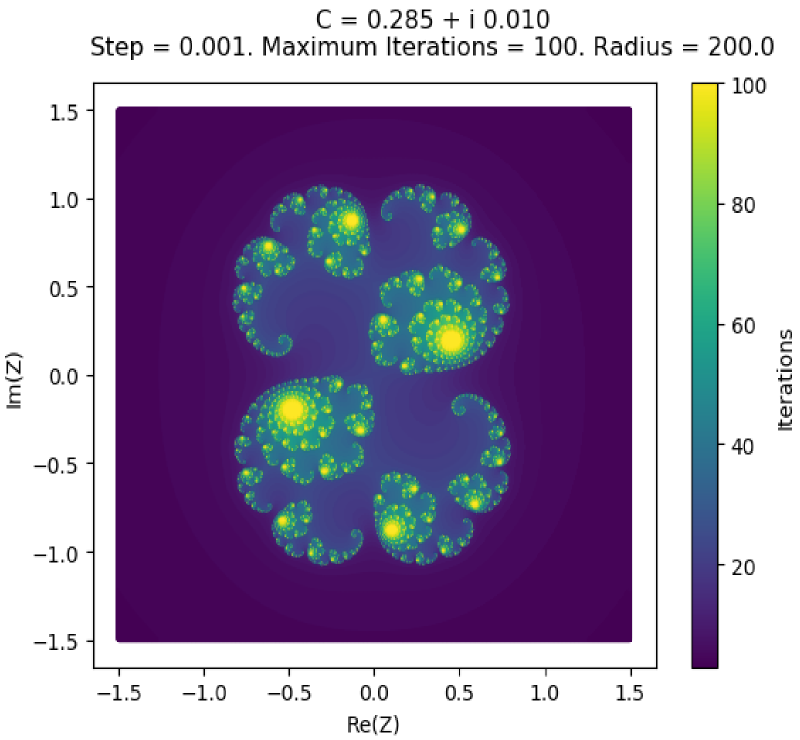

图 11: 彩色Julia图：$C = 0.285 + 0.010i, num = 100$

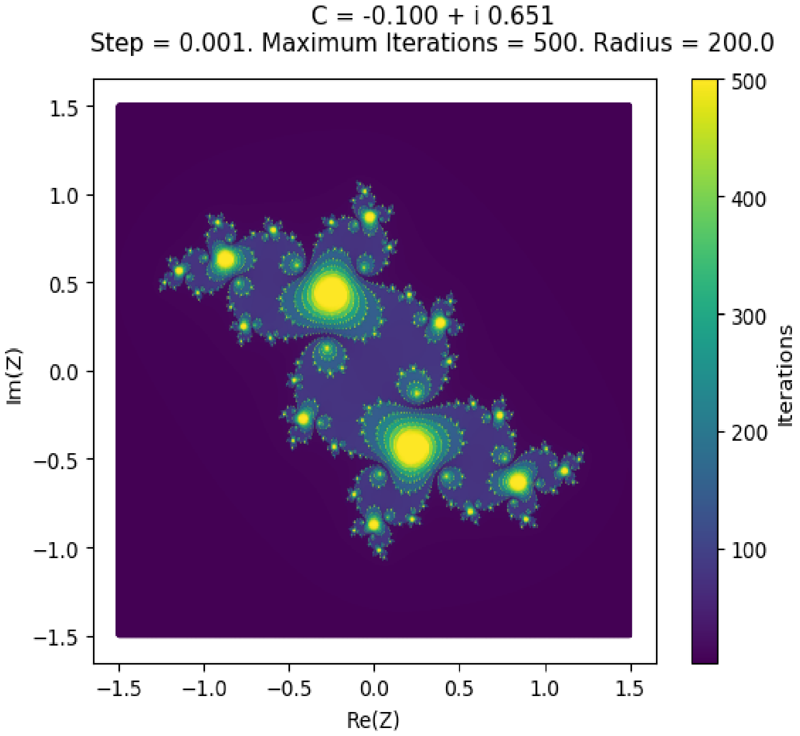

图 12: 彩色Julia图：$C = -0.100 + 0.651i, num = 500$

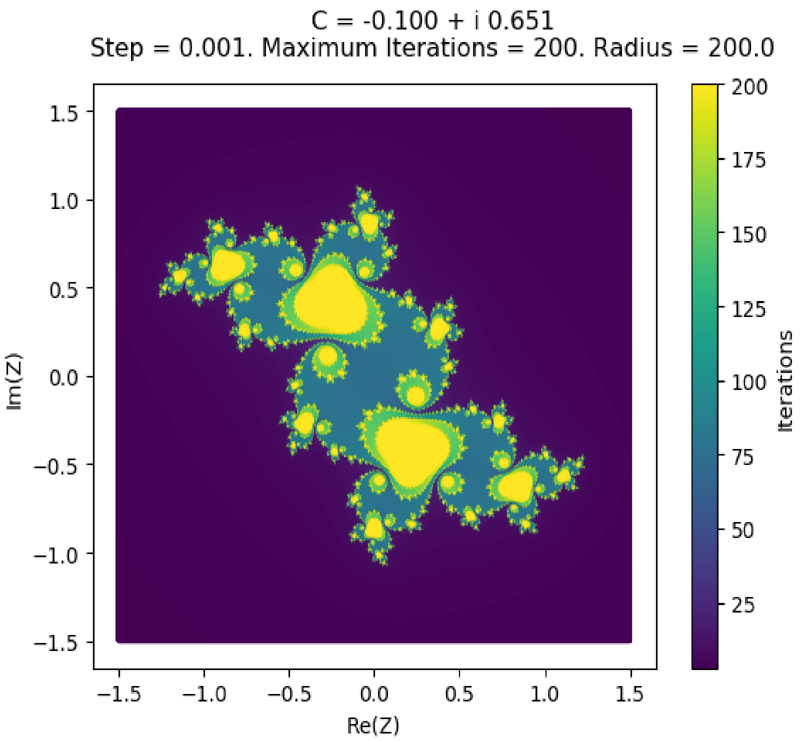

图 13: 彩色Julia图：$C = -0.100 + 0.651i, num = 200$

迭代次数num可以影响图片的明暗度与对比度，因为迭代次数多了，图片将更精确，也更接近真实

的Julia图。这点从图12与图13的对比中可以看出。迭代次数多的图12中间的分层更加明显，而图13则分层很不明显，只是黄色的一圈。

5 讨论

5.1 原理的讨论

运用迭代法求出Z_n的具体表达式，来判断其是否属于Julia集，是一种行之有效的方法。

但由于我们无法真正实现迭代无穷次，也无法求出无穷处的解析值，因此我们求得的Julia集并不是一个真正准确的Julia集。不过可以近似认为与真正准确的Julia集之间没有区别，因此画Julia集也是迭代法的一个非常重要的应用。

5.2 算法的改进

由于本人的计算机知识并不是很丰富，下面的讨论可能会欠妥，源代码的编写也肯定会有很多需要改进的地方，希望以后学习中能够不断完善。

我们设置for循环条件的时候，每步结尾的步骤如果设置成$x = x + step$和$y = y + step$则会出现很大的错误。虽然对大部分C的取值，这个错误对Julia集的图案无关紧要。但对于某些值，例如$C = 0$，将无法画出一个圆。例如我们如果设置范围为2，步长为0.01，则生成的图像上没有一个点，意味着没有一个点在Julia集内。

这是因为double型变量存储的精度问题。不停累加很小的量，将会导致存储误差的放大。

例如对于$C = 0$的情况，我们知道$x = -1, y = 0$属于Julia集。但循环找到这个点的时候，我们需要坐标是精确的$x = -1, y = 0$，实际上有微小偏差。这个微小偏差将导致迭代多次后Z_n大于迭代距离，程序认为这个点不属于Julia集。

此前，我发现$C = 0$时图上居然没有一个点，但花了很长时间也未找到算法的错误。后来意识到这个问题，将步长叠加的语句改成$x = -range + i * step$才得到正确的结果。

生成数据的程序通过for循环来选取点，画图的程序也利用for循环来判断点是不是在Julia集内或者点的颜色，都会使程序速度非常慢。目前我没有想到什么特别好的解决措施。不过与同学讨论后得知，可以通过某种函数直接对某一块像素点来着色，而不是给一个点的坐标来画散点图。这样不用通过for循环来选择点的横、纵坐标，输出的数据也不用输出大量的坐标，大大提升了程序运行速度。同样画图程序也不必读取大量的横、纵坐标数据，也可以直接通过着色而不是画散点图来画彩色Julia图。

5.3 结果的讨论

由结果可以得知，很多因素都将影响图片的样貌。最显然的就是设置的范围和步长，会影响图片的像素和大小，而C的选取，会决定图片的具体样貌。

其次，最大迭代次数的设置，也会直接影响图片的样貌。对于黑白Julia集图，会直接影响图上点的分布。例如图1和图2所示，一个是散点图，而另一个是比较“准确”的圆。对于彩色Julia集图，则会影响图片的对比度、明暗度等。如图12和图13，可以看到迭代次数多的图12中间的分层更加明显，而图13则分层很不明显，只是黄色的一圈。

我们可以通过图片直观地看出许多Julia集的性质，例如对称性、自相似性等。我们也可以通过设置C的取值来直观看出C对Julia集的影响。例如共轭的C的Julia集关于虚轴对称。而实部的相反性则并不

对应对称。如下面的图14、图15、图16所示。

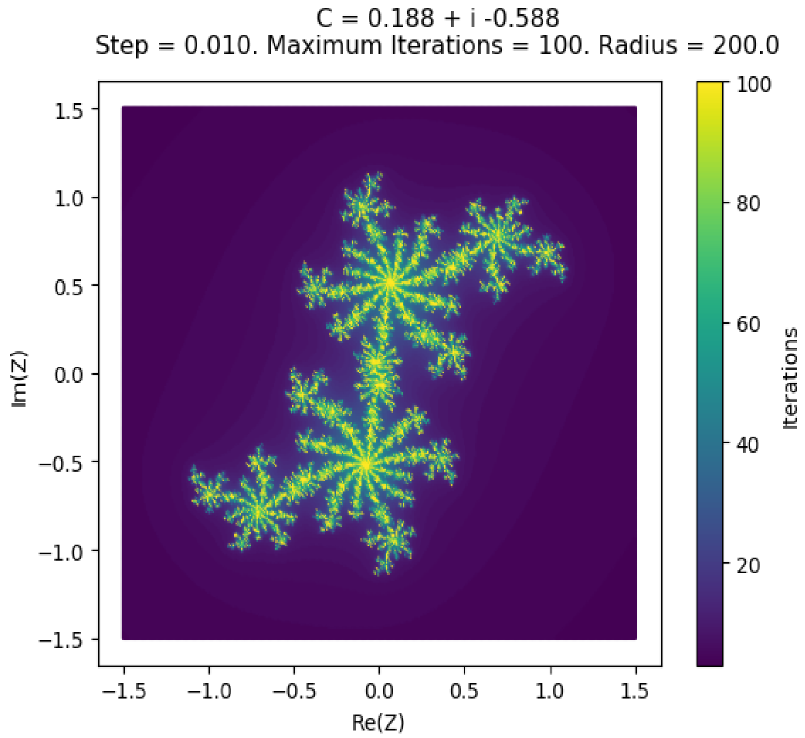

图 14: 彩色Julia图：$C = 0.188 - 0.588i, num = 100$

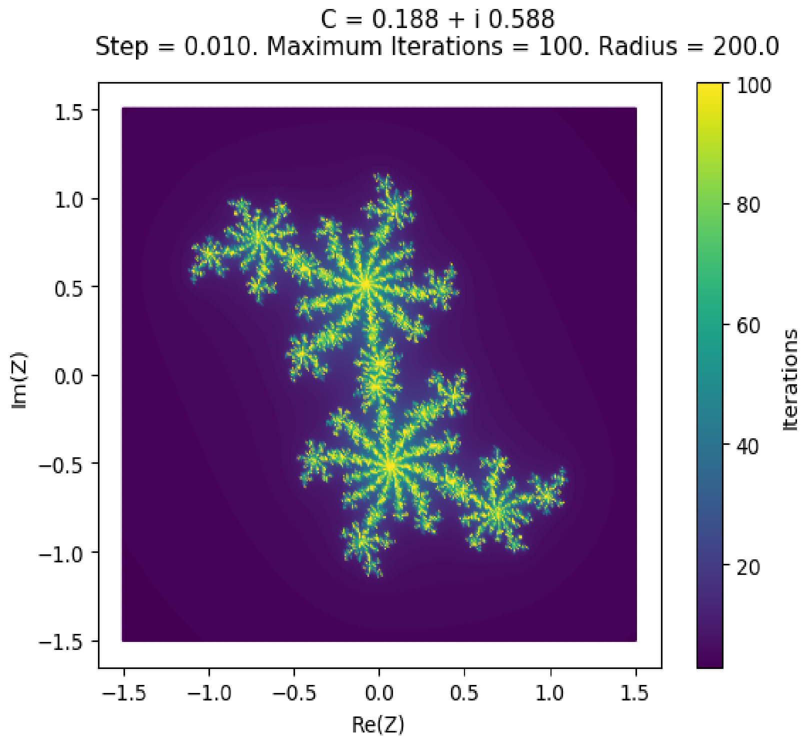

图 15: 彩色Julia图：$C = 0.188 + 0.588i, num = 100$

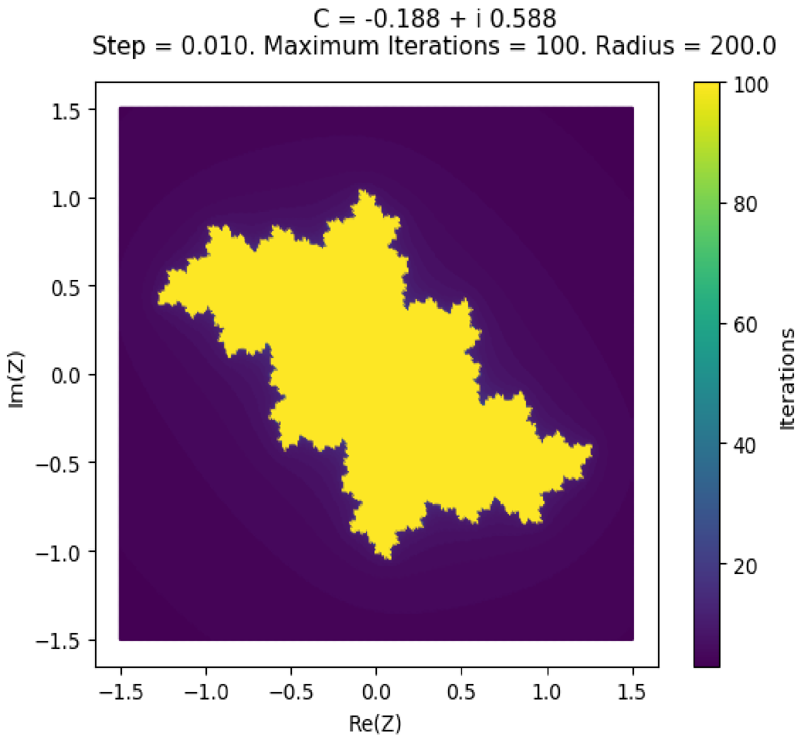

图 16: 彩色Julia图：$C = -0.188 + 0.588i, num = 100$