The Mandelbrot Set

The Mandelbrot set, the topic of this notebook, became famous as a simple model which produces extraordinarily complicated (and "beautiful") fractal structures. It is defined as the set of all points in the complex plane, (c_x, c_y) such that the complex map

$z \to z^2 + c$

i.e. $z_{n+1} = z_n^2 + c$, does *not* escape to infinity starting from z = 0. As we shall see it is related to the logistic map, and can be thought of as a generalization of the logistic map to the complex plane.

The code below is used to generate a Mandelbrot set. It takes as arguments the real and imaginary parts (**cx**, **cy**) of the complex number **c**, and returns (minus) the number of iterations that the map takes to escape (in practice to get to |z| > 2, because one can show that a point will escape to infinity if it once gets to a value where |z| > 2) up to a maximum of **lim**. Note the use of the W**hile** command (a similar programming style to that used in C).

In[1]:=
```
Clear["Global`*"]
```

In[2]:=
```
mandelbrot[cx_, cy_, lim_] := (
              z = 0; ct = 0;
              While[Abs[z] < 2.0 && ct <= lim,
                    z = z^2 + cx + I cy;
                    ++ct
                   ];
              -ct
         );
```

For example,

In[3]:=
```
mandelbrot[0.5, 0.5, 50]
```

Out[3]=
```
-5
```

so starting from c = 0.5 + 0.5 I, it takes 5 iterations for the point z = 0 to escape to |z| > 2.

Next we write the function as a module, rather than just a function, so that local variables, ct, and z, can be defined in the routine independentally of whether any variables of the same name exist in the rest of the *Mathematica* session. Note that the local variables can be initialized at the same time as they are declared, and we do this here:

In[6]:=
```
Clear[mandelbrot];
mandelbrot[cx_, cy_, lim_] := Module[{z = 0, ct = 0},
              While[Abs[z] < 2.0 && ct <= lim,
                    z = z^2 + cx + I cy;
                    ++ct
                   ];
              -ct
         ];
```

This routine is quite slow. Next we compile it so that it runs faster. To compile a function **f[x_] = x Log[x]** and call the compiled version **fc**, we write **fc = Compile[{x}, x Log[x]]**, where the quantity(ies) in curly brackets is (are) the argument(s). (The notation is similar to that for a pure function, for which we could write **f = Function[{x}, x log[x]]**). The following function is the compiled version of the above Module:

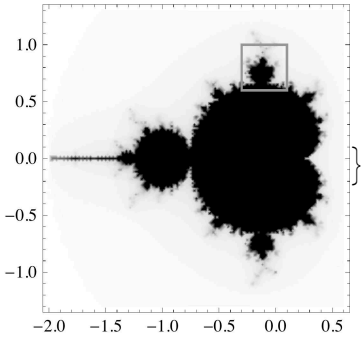
In[8]:=
```
mandelbrotC = Compile[{cx, cy, lim},
        Module
        [{z = 0. + I 0., ct = 0},
             While [Abs[z] < 2.0 && ct <= lim,
                    z = z^2 + cx + I cy;
                    ++ct
               ];
             -ct
        ]
];
```

Note that the arguments appear on the RHS of the function definition, rather than on the LHS as in the uncompiled version.

Below we use show the whole of the Mandelbrot set with a coarse resolution and compare the timing of the compiled and uncompiled version. The value of the function being plotted is characterized by the degree of blackness in a **DensityPlot**. The smaller the number the blacker it is. Since we compute the *negative* of the number of iterations, the blackest region is where the number of iterations is greatest.

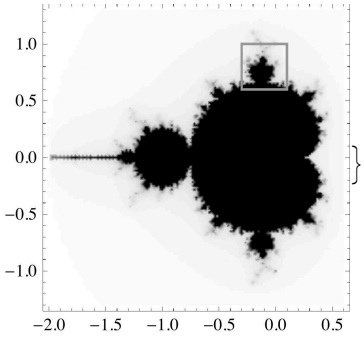
In[9]:=
```
Timing[DensityPlot[mandelbrot[x, y, 100], {x, -2, 0.6}, {y, -1.3, 1.3}, PlotPoints → 40,
  Mesh → False, ColorFunction → GrayLevel, Epilog → {{GrayLevel[0.6], Thickness[0.008],
     Line[{{-0.3, 0.6}, {-0.3, 1}, {0.1, 1}, {0.1, 0.6}, {-0.3, 0.6}}]}}]]
```

Out[9]=
```
{0.901516,
```



In[10]:=
```
Timing[DensityPlot[mandelbrotC[x, y, 100], {x, -2, 0.6}, {y, -1.3, 1.3}, PlotPoints → 40,
  Mesh → False, ColorFunction → GrayLevel, Epilog → {{GrayLevel[0.6], Thickness[0.008],
     Line[{{-0.3, 0.6}, {-0.3, 1}, {0.1, 1}, {0.1, 0.6}, {-0.3, 0.6}}]}}]]
```

Out[10]=
```
{0.176187,
```



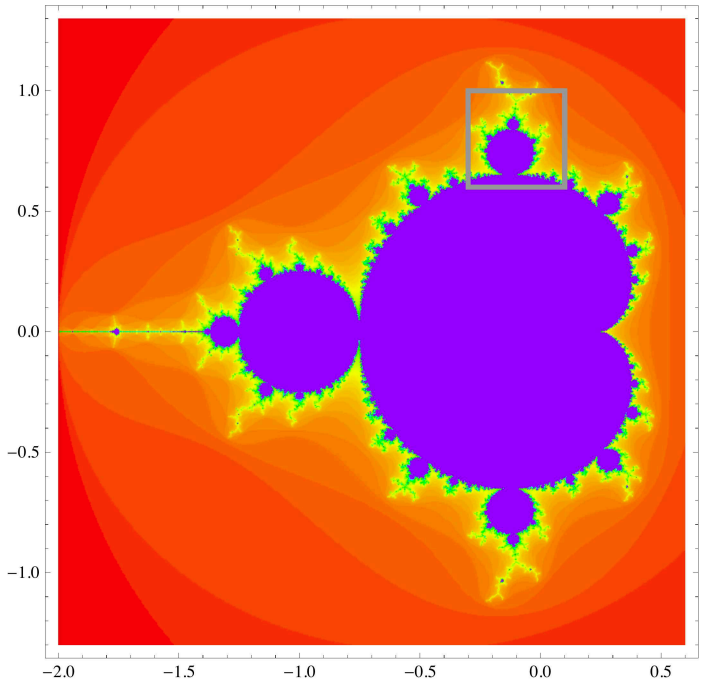
Note that the compiled version is about 5 times faster.

Remember that the darker the region the more iterations were needed to escape.

Now we use the compiled version to show the set with finer resolution and in color, using the ColorFunction option of **DensityPlot**. The way we call the Hue function is an example of a *Mathematica* "pure function". Areas in blue need a large number of iterations to escape while those in red need rather few. (Note that I plotted minus the result of the **mandelbrotC** function.)

In[11]:=
```
DensityPlot[-mandelbrotC[x, y, 200], {x, -2, 0.6}, {y, -1.3, 1.3},
 PlotPoints → 200, Mesh → False, ColorFunction → (Hue[0.75 #1^0.6] &),
 Epilog → {{GrayLevel[0.6], Thickness[0.008],
    Line[{{-0.3, 0.6}, {-0.3, 1}, {0.1, 1}, {0.1, 0.6}, {-0.3, 0.6}}]}}]
```

Out[11]=



This is the famous Mandelbrot set. Even at this resolution it is amazingly complicated. However, the full depths of its complexity can only be appreciated when we blow up different regions of it, which we will do shortly.

Notice that the largest part of the figure is in the shape of a cardioid out of which grow many "buds" of different sizes. The largest bud, to the left of the cardioid joins it at one point, x=-3/4, y = 0. The "cusp" in the cardoid on the right is at x = 1/4, y = 0. Clearly if c is real then, since we start off with z real ($z_0 = 0$), all subsequent values $z_n$must also be real. There is actually a strong connection between the map used to generate the Mandelbrot set, $z_{n+1} = z_n^2 + c$, specialized to real c (and z), and the logistic map $x_{n+1} = 4\,\lambda\,x_n\,(1 - x_n)$. To see this let $z_n = a\,x_n + b$ and verify that the two maps are the same if b = 2λ and a = -4λ with c then related to λ by

$c = 2\,\lambda\,(1 - 2\,\lambda)$.

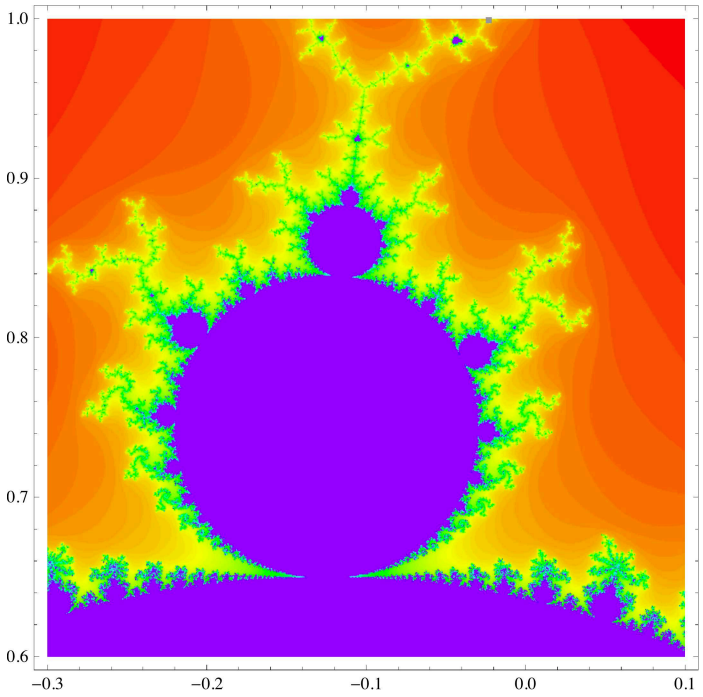
Hence, as λ varies from 1/4 to 1 (all the interesting region of the logistic map except for the trivial part with a fixed point at x = 0) c varies from 1/4 to -2. The fixed point region of the logistic map, 1/4 $< \lambda <$ 3/4 corresponds to 1/4 > c > -3/4, which is precisely the region of the big cardioid. The period two limit cycle, 3/4 < λ < 0.8626 corresponds to the largest bud the left of it, the period 4 limit cycle to the bud growing out to the left of that and so on. Additional "blobs" further to the left on the x axis, and which appear separated from the main region (though they are not) correspond to islands of limit cycle behavior in the logistic map, which, as you will recall, period double their way to chaos.

There is therefore a connection between the Mandelbrot set and the logistic map. However, the full richness of the Mandelbrot set is only seen by going into the complex plane.

Now we will blow up different regions of the set, starting with an enlarged the bud at the top of the cardioid (the area in the grey box in the figure above)

In[12]:=
```
DensityPlot[-mandelbrotC[x, y, 200], {x, -0.3, 0.1}, {y, 0.6, 1},
 PlotPoints → 200, Mesh → False, ColorFunction → (Hue[0.75 #1^0.63] &),
 Epilog → {{GrayLevel[0.6], Thickness[0.008], Line[{{-0.0234, 0.99875}, {-0.0234, 0.99925},
     {-0.0229, 0.99925}, {-0.0229, 0.99875}, {-0.0234, 0.99875}}]}}]
```

Out[12]=

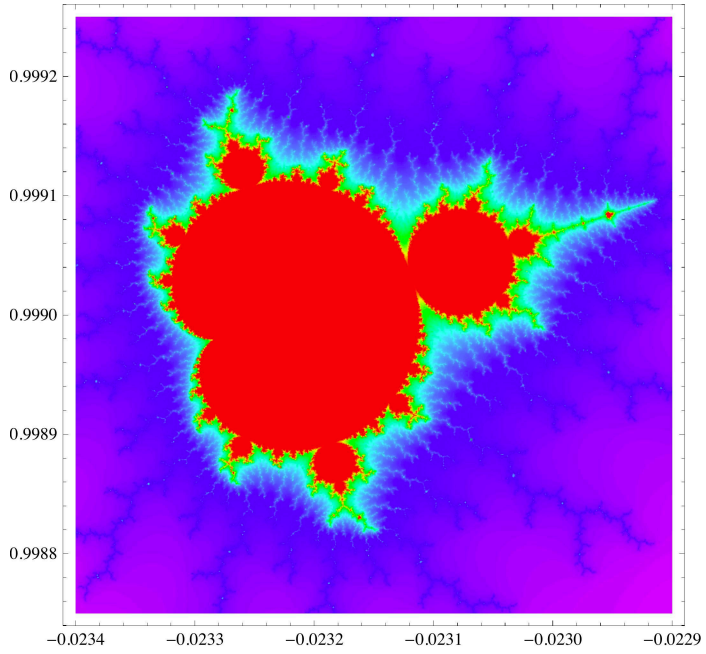


Notice how the same structure is repeated in many places and in many different scales. This is characteristic of a fractal structure.

Below we show a tiny region at the top edge in the above around x=-0.023, y = 1. (It is marked by a grey box in the above figure but it is so small it is hard to see.) In order to bring out the details of the structure we have increased the maximum number of iterations to 500.

In[13]:=
```
DensityPlot[mandelbrotC[x, y, 1000], {x, -0.0234, -0.0229}, {y, 0.99875, 0.99925},
 PlotPoints → 200, Mesh → False, ColorFunction → (Hue[0.8 #1^7] &)]
```

Out[13]=

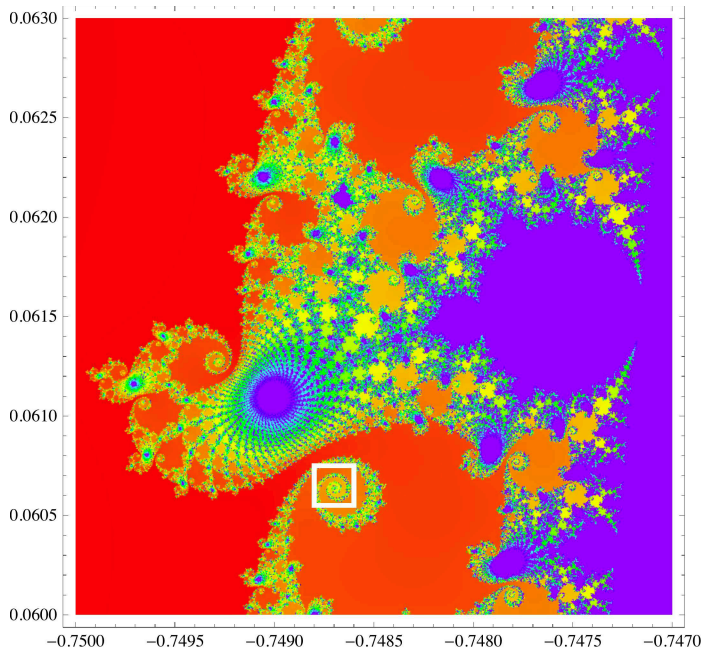


We find the Mandelbrot set again! Notice how expanded the scale is. This miniature Mandelbrot set is actually connected to the rest of the set, but the connecting filaments are too fine to be seen on this plot.

Next we look at a greatly enlarged view of the edge of the cardioid in the first high resolution figure (3rd figure overall), in the notch near where it joins the bud at the left. The scale is nearly a thousand times greater than in the first figure.

In[14]:=
```
DensityPlot[-mandelbrotC[x, y, 1000], {x, -0.75, -0.747}, {y, 0.06, 0.063},
 PlotPoints → 200, Mesh → False, ColorFunction → (Hue[0.75 #1] &),
 Epilog → {{GrayLevel[1.], Thickness[0.008], Line[{{-0.7488, 0.06055}, {-0.7488, 0.06075},
      {-0.7486, 0.06075}, {-0.7486, 0.06055}, {-0.7488, 0.06055}}]}}]
```

Out[14]=

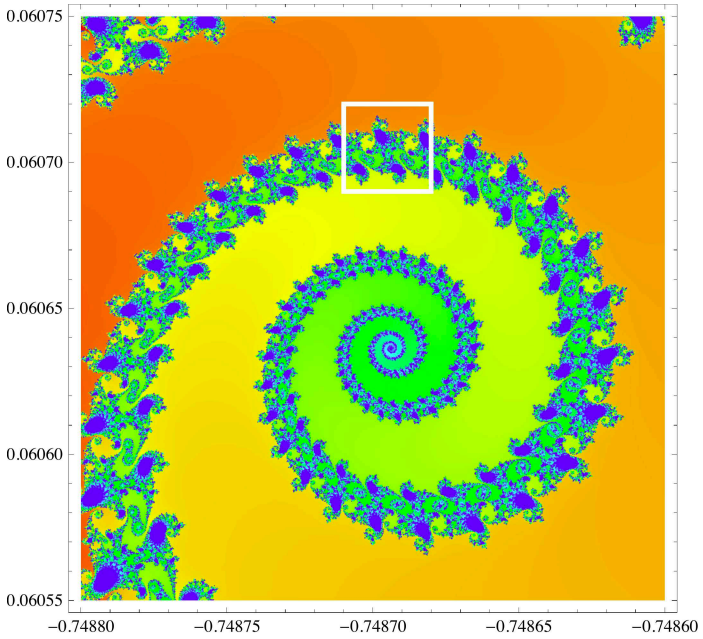


Notice the complicated spiral structures.

Suppose we now blow up the center of the spiral arm near the bottom and slightly to the left. What will be see?

In[15]:=
```
DensityPlot[-mandelbrotC[x, y, 500], {x, -0.7488, -0.7486}, {y, 0.06055, 0.06075},
 ColorFunction → (Hue[0.7 #1] &), PlotPoints → 200, Mesh → False, Epilog →
  {{GrayLevel[1.], Thickness[0.008], Line[{{-0.74871, 0.06069}, {-0.74871, 0.06072},
      {-0.74868, 0.06072}, {-0.74868, 0.06069}, {-0.74871, 0.06069}}]}}]
```

Out[15]=

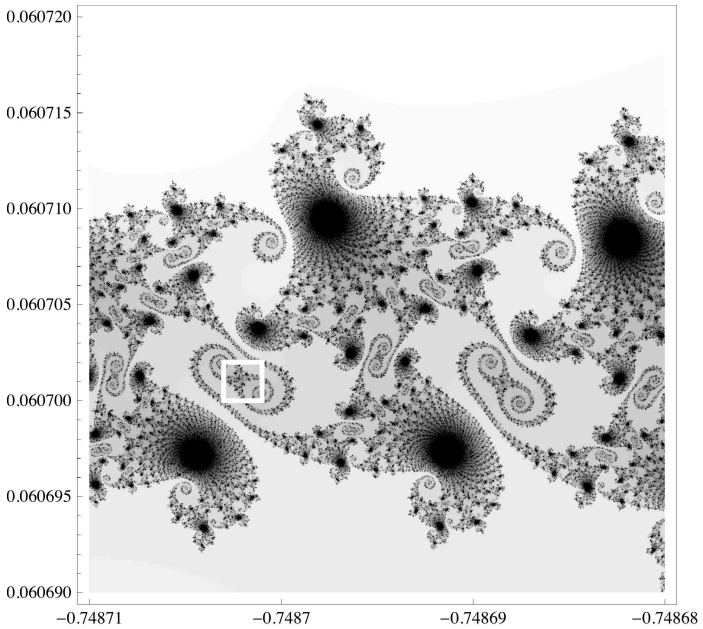


There is an enormous amout of detail in the spiral arm.

Let's continue to increase the scale and enlarge a section of the arm, at the highest point of the spiral.

In[16]:=
```
DensityPlot[mandelbrotC[x, y, 1000], {x, -0.74871, -0.74868},
 {y, 0.06069, 0.06072}, PlotPoints → 200, Mesh → False, ColorFunction → GrayLevel,
 FrameTicks → {{-0.74871, -0.7487, -0.74869, -0.74868}, Automatic, None, None}, Epilog →
  {{GrayLevel[1.], Thickness[0.008], Line[{{-0.748703, 0.0607}, {-0.748703, 0.060702},
      {-0.748701, 0.060702}, {-0.748701, 0.0607}, {-0.748703, 0.0607}}]}}]
```

Out[16]=

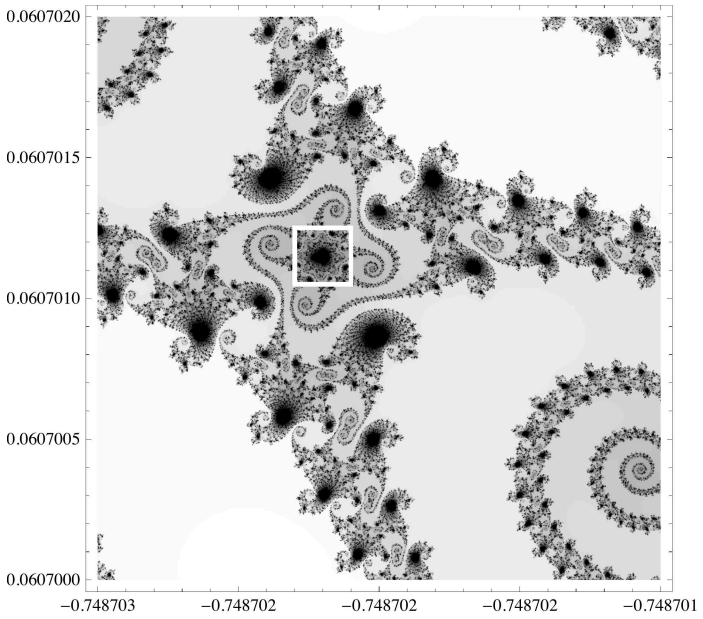


We have now blown up the figure of the entire Mandelbrot set at the top by a factor of about 10^5 and we see intricate details down to the smallest scale visible.

Next let's blow up one of the two regions where a thin loopy strand splits into two spirals turning in opposite directions. We take the one near x=-0.748702, y = 0.060701.

In[17]:=
```
DensityPlot[mandelbrotC[x, y, 1000], {x, -0.748703, -0.748701},
 {y, 0.0607, 0.060702}, PlotPoints → 200, Mesh → False,
 ColorFunction → GrayLevel, Epilog → {{GrayLevel[1.], Thickness[0.008],
    Line[{{-0.7487023, 0.06070105}, {-0.7487023, 0.06070125},
      {-0.7487021, 0.06070125}, {-0.7487021, 0.06070105}, {-0.7487023, 0.06070105}}]}}]
```

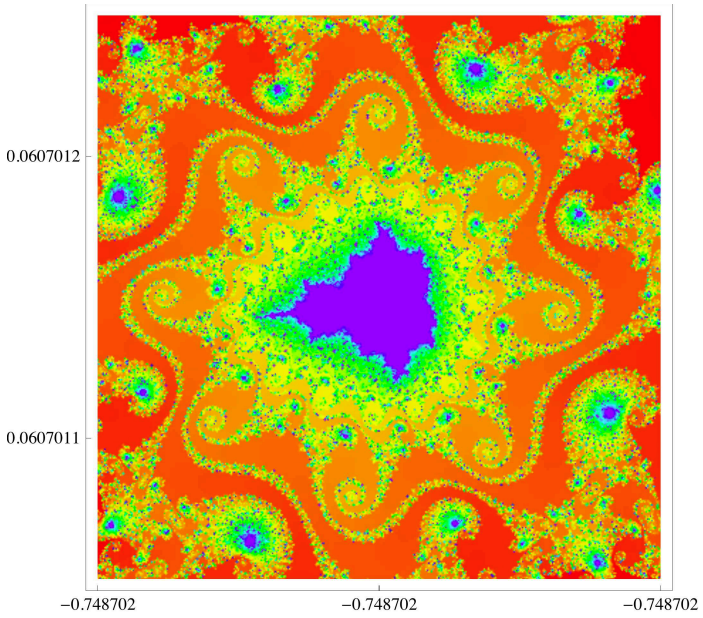
Out[17]=



Finally we zoom in on the "blob" at the center of the figure (surrounded by 4 spirals).

In[18]:=
```
DensityPlot[-mandelbrotC[x, y, 2000],
 {x, -0.7487023, -0.7487021}, {y, 0.06070105, 0.06070125},
 ColorFunction → (Hue[0.75 #1] &), PlotPoints → 100, Mesh → False,
 FrameTicks → {{-0.7487023, -0.7487022, -0.7487021}, {0.0607011, 0.0607012}, None, None}]
```

Out[18]=



Lo and behold, the Mandelbrot set again! Note that the scale of this figure is about 10^7times greater than in the first figure we showed of the Mandelbrot set.

Julia Set

The Julia set is related to the Mandelbrot, set since it also involves the study of the escape to infinity of a map, z -> f(z) involving a parameter c, but rather than fixing the initial point to be z = 0 and considering the set of all values of c for which z does not escape to infinity, we now consider the set of all initial points z that does not not escape to infinity for a *fixed* complex c. The form of the Julia set depends very much on the choice of the function f(z) and the parameter c.

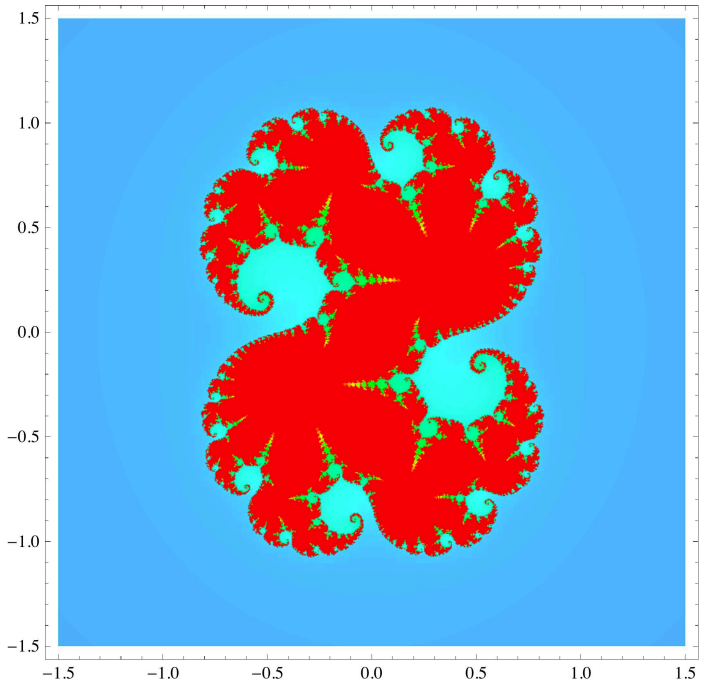
The code below iterates the quadratic map, z -> z^2 + c, the same as for the Mandelbrot set, and determines the number of iterations it takes for z to escape for the given initial value of z (= x + iy) and the (fixed) value of c (= cx + icy).

In[19]:=
```
juliaC = Compile[{x, y, lim, cx, cy},
         Module
         [{z, ct = 0},
              z = x + I y;
              While
                      [Abs[z] < 2.0 && ct <= lim,
                    z = z^2 + (cx + I cy);
                    ++ct
                ];
              ct
         ]
];
```

The plot below show the Julia set for a particular value of c.

In[20]:=
```
DensityPlot[-juliaC[x, y, 200, 0.27334, 0.00742], {x, -1.5, 1.5},
 {y, -1.5, 1.5}, ColorFunction → (Hue[0.55 #1] &), PlotPoints → 200, Mesh → False]
```

Out[20]=



We now show the central region of the above plot enlarged.

In[21]:=
```
DensityPlot[-juliaC[x, y, 50, 0.27334, 0.00742], {x, -0.5, 0.5},
 {y, -0.5, 0.5}, PlotPoints → 200, ColorFunction → (Hue[0.75 #1] &), Mesh → False]
```

Out[21]=