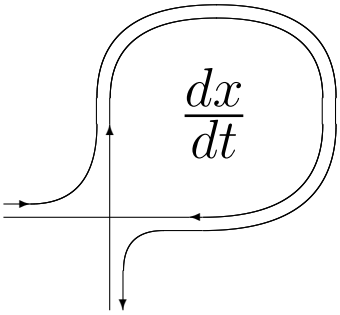


DIFFERENTIAL EQUATIONS
AND
CONTROL PROCESSES
N 4, 2005
Electronic Journal,
reg. N P23275 at 07.03.97

http://www.neva.ru/journal
e-mail: diff@osipenko.stu.neva.ru

Ordinary differential equations

METHODS AND APPLICATIONS TO DISPLAY QUATERNION JULIA SETS

Alessandro Rosa

Freelance programmer
c/o Locatelli,
Via Cappuccini 116/A,
I-72100 Brindisi, Italy
e-mail: zandor_zz@yahoo.it

Abstract

After explaining the details of an efficient technique, developed in early 1980s, to plot digital images of quaternion filled-in Julia sets, later considerations on it will introduce a new modification for displaying such sets with no interior too. A step by step description via *pseudo*-C++ is given. The reader is assumed to be familiar with advanced programming and quaternion calculus.

1 History

In the turn of 1970s and 1980s, consequently to the worldwide revival of interests in the theory of holomorphic dynamics[1], digital imaging started to play a relevant role in this field thanks to the detailed pictures offered; more widely

[1]This term gathers the iterations of functions in one and in several complex variables.

speaking, this same impulse contributed to definitely show how computers can be regarded as essential to aid experimental mathematics so that, besides the developments in the related analysis, rendering methods were produced for highly detailed images of these Julia sets.

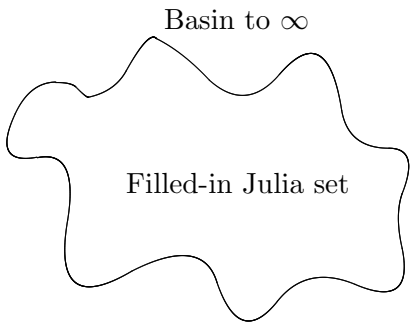


Figure 1.1: The typical topological configuration for a filled-in Julia set.

Achievements first stepped in $\mathbb{C}$. Not casually: this was the original numerical field where local holomorphic dynamics arose during 1870-80s with Schröder and Koenigs. During 1910-20s, global results came from the pioneering works by Fatou and Julia. Along few developments over later decades, holomorphic dynamics finally reached 1980s when, on a revival wave, new enthusiasms drove efforts to resume deep and old results: for computer imagery, this trend begun with plots in $\mathbb{C}$. In 1982 Alan Norton definitely opened to quaternions [14] and implicitly showed that, contrary to the same graphics computed for complexes, clear views of Julia sets require to be rendered via ray-tracing here: otherwise they[2] would be flat looking and their complicated topological structure would not be evinced. During 1983-87 John Holbrook's works [8, 9] followed, where the inverse iteration, exported from $\mathbb{C}$, was also tried. In 1989-90 Norton came back [15] to the topic. We also mention the collaborative results [6, 7] by Hart, Kauffman and Sandin, developing either Holbrook's and Norton's results together with a more refined ray-tracing as well as some additional considerations on how to improve that inverse method. In 1996, on the front of Algebra, Bedding and Briggs [2] showed that differentiation for quaternions is less obvious as for reals and complexes, because of susceptible to different approaches[3]; they also focused on related consequences in iteration theory, such as the generation of the Mandelbrot set $\mathcal{M}_{\mathbb{H}}$ for the classic quaternion quadratic iterator

$$h^2 + c \qquad\qquad (h, c \in \mathbb{H}) \tag{1.1}$$

and on 'regularly iterable linear quaternion maps', i.e. quaternionic functions whose iterates preserve regularity under iteration. Latest results, for dynamics of quaternion maps, are due to Jazek and Petek [10], by Lakner and Petek [11] and by Petek [16] during mid 1990s, where either analytical and dynamical properties are deepened: for example, one mentions the 'equator' concept,

[2]Existing in 4 dimensions, they require to be submersed from $\mathbb{R}^4$ to $\mathbb{R}^3$, or to $\mathbb{R}^2$, for drawing purposes.
[3]See also Cullen-Rinehart's [3, 17], Fueter's [5], Sudbery's [18].

already noted by Holbrook in late 1980s and referring to the distribution, inside $\mathcal{M}_{\mathbb{H}}$ for (1.1), of Julia sets intersecting the imaginary axis at least twice or more often. In the same spirit as original Norton's, one sees a recent software production, devoted to display Julia sets in higher dimensions, such as for hyper-complex numbers and octonions.

2 Basics

2.1 The pixel grid

The computer generation of Julia sets in $\mathbb{C}$ is mainly based upon a 'walk' along the screen rows and columns. For each screen point P, one defines an ordered and unique pair of coordinates (x, y), associated to the complex $\rho = x + iy$. Let f be a given map, so the goal is to set a color, related to the value of the n-fold iterate $\rho_n = f^{\circ n}(\rho)$, at P. This algorithm resumes below into *pseudo*-C++ code. The two nested `for` loops allow to walk through the screen:

```
for ( int y = 0 ; y < h ; y++ )
{
    for ( int x = 0 ; x < w ; x++ )
    {
        find the complex value z
        of the ordered pair (x,y);
        iterate(z) ;
        plot(x,y,color) ;
    }
}
```

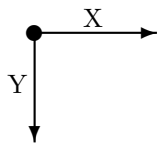


Figure 2.1: **The Walk.** It starts from the top left corner $(0, 0)$, according to the standard computer display, up to the bottom right point $(w - 1, h - 1)$.

Let also `w` and `h` be the screen *width* and *height* respectively. This process can be exported to quaternion Julia sets $\mathcal{J}_{\mathbb{H}}$, but at the cost of some proper changes. A quaternion $q \in \mathbb{H}$ ([4]) is a wider numerical extension of a complex value and consists of 4 components[5] $r, m, n, p \in \mathbb{R}$. Then one has:

$$q = r + im + jn + kp \tag{2.1}$$

where $i^2 = j^2 = k^2 = -1$ and multiplication is non-commutative; so quaternions are not an abelian group.

[4]The letter 'H' refers to the first letter of 'Hamilton', the inventor of quaternions.
[5]From latin '*quater*' : four.

3 The algorithm

3.1 Tips for the 3-D model

How can 4 *coordinates fit the 3-D rendering model of a quaternion Julia set ?*

From a mathematical viewpoint, these sets cannot show fully up in $\mathbb{R}^4$, so digital pictures of submersions to $\mathbb{R}^2$ or $\mathbb{R}^3$ have to be produced. $\mathbb{R}^3$ was the most widespread solution adopted: each screen point will be displayed according to any triplet within the 4 components; these 3 values are set as X-Y-Z coordinates. While walking through the axes, one matches $\mathbb{R}^3$-points to quaternions q (2.1); the remaining fourth vector is *initially* set to 0. Every q is now ready to be iterated and colored. Situation slightly complicates as one moves later from 3-D to the 2-D screen. The two points below refer to the simplest approach, which has to be dropped off for the considerable loss of numerical accuracy as one re-scales ...

1. ... the original 3-D model into the unit cube C, with side $l = 1$. Here old coordinates are *normalized*: i.e. one gets a common basis to re-scale the model to new arbitrary dimensions.

2. ... while moving from $\mathbb{R}^3$ to $\mathbb{Z}^2_+$ (the bi-dimensional computer screen with positive integer coordinates).

Best and fastest results come if these two steps are no longer regarded.

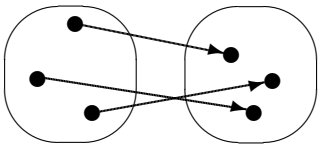


Figure 3.1: Injectivity.

A first efficient solution comes from basic considerations in General Topology. Given two topological spaces A, B, let $f : A \to B$ be a *one-to-one map* (or *injection*), so $f(x_1) \neq f(x_2)$ for any pair $x_1, x_2 \in A$ where $x_1 \neq x_2$. This new model will help to match each quaternion number (in 4-D) to a screen point (in 2-D). For not losing numerical precision in the way, one suggests to apply the reverse path from the screen to quaternion space:

$$\mathbb{Z}^2 \to \mathbb{R}^3 \to \mathbb{H};$$

the graphical aspects concern $\mathbb{Z}^2$ and $\mathbb{R}^3$ exclusively, while precision more involves with $\mathbb{H}$. Things will get far more interesting as one notices that the escape method [1] cannot be solely applied to display $\mathcal{J}_{\mathbb{H}}$ because it just colors those points whose orbit is trapped into the complement set $H_{\mathcal{F}} = H \backslash H_{\mathcal{J}}$: the 'Fatou set'. But the problem here is that, since several points of $\mathcal{J}_{\mathbb{H}}$ relate to the same iteration index by means of the escape method, they condense

into a cloud with a same color; thus details of $\mathcal{J}_{\mathbb{H}}$ will not come out and the index cannot be applied to define a color as in $\mathbb{C}$. A good solution is to apply an illumination model, like the efficient 'Phong': this is a good compromise between fast computations and an artificial light rendering with high quality.

3.2 Iterations

The example code below is one of the few elements in common with the complex case. The point q is iterated under $q^2 + c$ ($q, c \in \mathbb{H}$) and a test, if $q \in H_{\mathcal{J}}$ [6], is performed. In computational terms, the `while` loop arrests as the orbit escapes the test ball of given radius or when the maximal iterative index is attained to prevent endless looping. This function returns a boolean value in order to know if conditions, for q to be plot on the screen, are met.

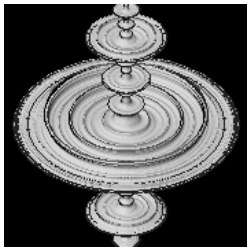


Figure 3.1: Rotated slice of the filled-in Julia set for $h \mapsto h^2 - 1.5$.

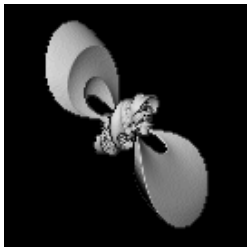


Figure 3.2: A '*papillon*' Julia set for $h \mapsto \frac{1}{h^2} + q, q = 0.19 - 0.5i + 0.1k$.

```
BOOL quIterate( quaternion q, quaternion c,
                int& count, double& length,
                double radius, int comp_limit )
{
    quaternion next_q ; // next point in the orbit
    count = 0 ;         // set the iteration index for start
    BOOL bDraw = TRUE ;

    while ( bDraw )
    {
        length = qu.abs() ; // computes the absolute value
        next_q = q * q + c ;
        count++ ;           // the iterative index

        if ( length > radius ) return bDraw = FALSE ;
        else if ( count == comp_limit ) break ;
        else q = next_q ;
    }
```

[6] In this topological configuration, the orbit does not escape to ∞.

```
    return bDraw ;
}
```

3.3 Plotting the screen

Assume the screen at the *logical* machine level: a grid of pixels, where each element is uniquely defined by one ordered pair of coordinates (x, y). Let the quaternion hyper-cube be an ideal 3-D metric model of the geometric locus we are going to scan and find which seed points therein belong to $H_{\mathcal{F}}$. As one drops the fourth coordinate p and sets it always to 0, the Quaternion hyper-cube turns into a real 3-D cube and becomes an optimal descriptor for our purposes in geometrical terms exclusively: the resulting space is homeomorphic to any cube in $\mathbb{R}^3$, but quaternion calculus still holds here.

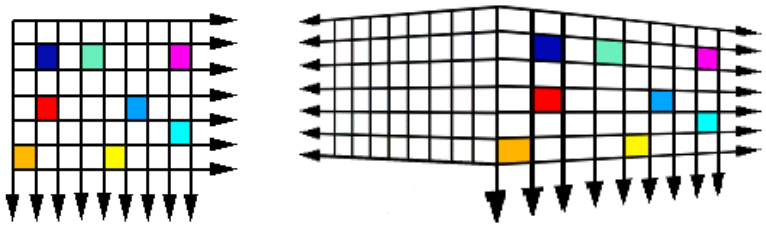


Figure 3.1: **Mapping onto the screen.** On the left, a matrix representing the screen. On the right, a perspective model of the cube. Colors denote the matching between 2-D and 3-D models.

The exploration needs both metrics and a way to run through the three dimensions. All this is supplied by a pair of minimum and maximum values per each cube side X-Y-Z. For example:

```
double x_interval = x_max - x_min ;
double y_interval = y_max - y_min ;
double z_interval = z_max - z_min ;
```

at this point, we also assume three *discretization* (`dx`, `dy`, `dz`) values for the x, y, z axes for locating any point within the quaternion cube.

```
double dx = x_interval / (double)window_client_area_width ;
double dy = y_interval / (double)window_client_area_height ;
double dz = z_interval / (double)discretization ;
```

So a one-to-one map from the screen onto the 3-D model is set. The walk can be now performed:

```
for ( int py = 0 ; py < img_h ; py++ )
{
    for ( int px = 0 ; px < img_w ; px++ )
    {
        x = x_min + (double)px * dx ;
        y = y_min + (double)py * dy ;
        z = z_min ;
        ...
    }
}
```

As in figure (3.1), the routine relates each a 2-D point to one pile of contiguous sub-cubes in the 3-D model; one runs through the pile until a point of $\mathcal{J}_{\mathbb{H}}$ is (possibly) met. If so, the run stops and one does not care of next sub-cubes[7]: they are discarded from computation since being invisible from the pile start (the screen surface, in our terms); in fact they are hidden by the color of the related screen point, like in figure (3.2).

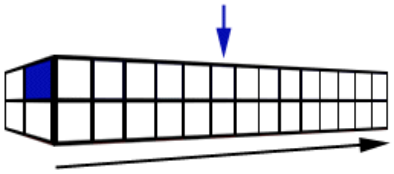


Figure 3.2: **Piles of visibility.** Screen and cube are ideally glued here to lessen the association as in figure (3.1). The black arrow indicates the scanning direction and the blue one refers to the found $\mathcal{J}_{\mathbb{H}}$-point within that same pile in the 3-D model.

3.4 The scanning direction

It consists in running from (`x,y,z = Zmin`) to (`x,y,z = Zmax`): so `z` ranges through the interval $\mid$ `Zmax` $-$ `Zmin` $\mid$. According to the discretization of the interval, one attains every single point step by step and tests if it belongs to the $H_{\mathcal{F}}$ by calculating the orbit via the function coded in section (3.2). If so, $\mathcal{J}_{\mathbb{H}}$ is met and one stops the scan for that pile. Otherwise `z` increments by `dz` again and again up to `Zmax`.

3.5 A prime Boundary Detection and refinements

Now the heart of the whole process. First we recall a definition from General Topology: for any subset A of a topological set X, the boundary ∂A is the set $\overline{A} \cap \overline{X \backslash A}$. See figure (3.2). In our terms, X and A are played by the quaternion cube and the filled-in $\mathcal{J}_{\mathbb{H}}$ respectively.

[7]They are assumed transparent.

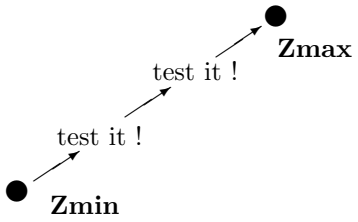


Figure 3.1: The testing walk.

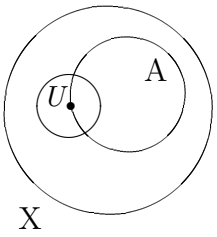
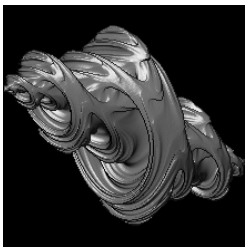


Figure 3.2: A boundary example.

Therefore the necessary and sufficient condition to detect $\mathcal{J}_{\mathbb{H}}$ is to find sufficiently small regions intersecting with more than one basin. An higher



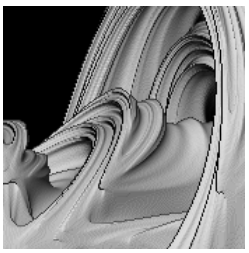


Figure 3.3: Slices of the Julia set for $h \mapsto h^2 - (0.55 + 0.55i)$.

discretization level assures a stronger graphic refinement, even if at the cost of huger computations. The approach relies on a two-steps boundary recognition, in order to find a sufficiently close neighborhood and set the ground for a next test-based upon analogous rules: actually one has to check if two contiguous points lie inside different basins. A first detection is quite easy. Since *the typical basins distribution for generic polynomials always assumes a region of convergence to* ∞, one can resume as follows:

1. *if* two contiguous points are seeds of bounded orbits to a same attracting fixed point, *then* they are inside the filled-in $\mathcal{J}_{\mathbb{H}}$;

2. *if* one point is seed of a bounded orbit, but the next one does not, *then* $\mathcal{J}_{\mathbb{H}}$ was crossed by the scanning path;

3. *if* both contiguous points are not seeds of bounded orbits, *then* they belong to the basin to ∞, i.e. outside the filled-in $\mathcal{J}_{\mathbb{H}}$;

In coding terms, boundary is detected as the logical `xor` operator returns `true`: at most one condition is to be met. In practice, one assumes a path running from `zmin` up to `zmax`, so that the routine `quIterate`, returning `false` until `z` is inside the basin to ∞, finally yields `true` as one enters the filled-in

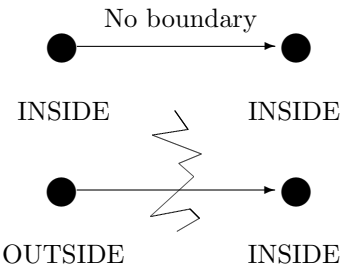


Figure 3.4: Meeting the boundary.

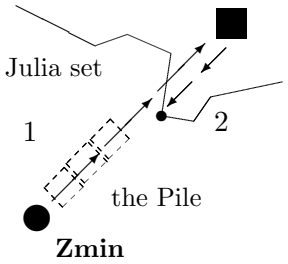


Figure 3.5: Crossing the Julia set (seeing through piles of pixels).

$\mathcal{J}_{\mathbb{H}}$ (see fig. (3.4)). As the boundary is trespassed, we get one step back to the previous point (see figure (3.5)). Here the second refinement process begins.

```
do{
    Take on three temporary x,y,z
    (actually only z would range over the interval,
     i.e. x, y are fixed per each pile);

    Set them into a quaternion for the further test ;
    Then iterate it !

    Is the n-th image point z_n trapped
    into the filled-in Julia set?

    NO: set the minimal bounding value of the interval at z_n ;
    YES: set the maximal bounding value of the interval at z_n ;
}while( z <= z_max );
```

Algorithm for a first boundary detection.

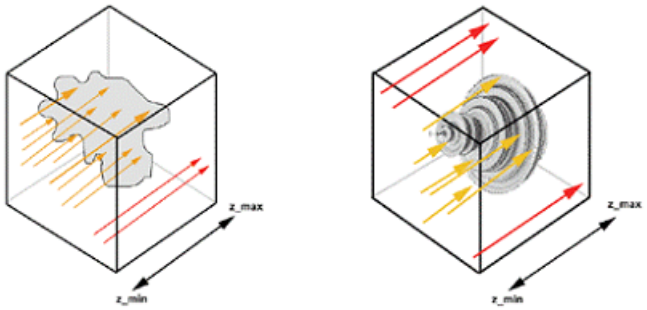


Figure 3.6: **Boundary Detection.** The points of the side, having `z = zmin`, are fired towards the set until they crash onto it. This process behaves as a sort of envelope and it models the 3-D space as the Quaternion Julia set shape.

Things slightly get more complicate (but even more appealing for programmers!) as the boundary locus is refined by turning the above routine into a

new recursive loop until the interval shrinks under a limit value, when the given accuracy is reached. The graphical discussion appears in fig. (3.7). Let `interval_dz` be our interval. Really one does not find the exact locus of $\mathcal{J}_{\mathbb{H}}$, but just very close points to it.

```
do{
    Take on three temporary x,y,z
    (really only z would range over
    the interval; i.e. x, y are fixed);

    Set them into a quaternion for the further test;
    Then iterate it !

    Is the n-th image point z_n trapped
    into the filled-in Julia set?

    NO: set the minimal bounding value of the interval at z_n ;
    YES: set the maximal bounding value of the interval at z_n ;

    ADDITIONAL CODE:
    ... then shrink interval_dz somehow;
}while( test if interval_dz is sufficiently
        wide to keep this code in progress ) ;
```

The refinement algorithm.

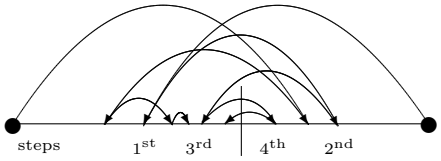


Figure 3.7: A scheme resuming the algorithm work to progressively refine the position of the boundary locus by changing the ends of the test interval.

Figure 3.8: A quaternion 'Dendrite' for $h \mapsto h^2 + i$ and two blow-ups.

3.6 Turning on the light

Since we adopted a 3-D model to draw $\mathcal{J}_{\mathbb{H}}$, an illumination model is required. We will not explain the technical details, because it is not a goal of this work (see [4]). The Phong model was adopted here because of granting good results and fast computations. Lambertian approach was also tried, but we do not finally push for it because details[8] are not sufficiently enhanced (see (3.1, 3.2)).

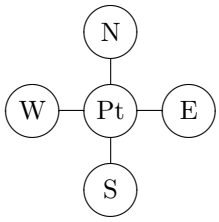


Figure 3.1: The four movements from a given point.

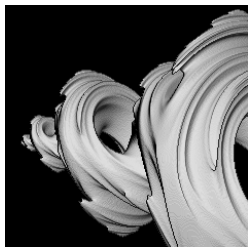


Figure 3.2: Slice of the Julia set for $h \mapsto h^2 + q, q = -0.8 + 0.4i$.

Suggestions will be given. As a $\mathcal{J}_{\mathbb{H}}$-point is found, take on 4 neighboring points (see fig. (3.1)). Each point of the neighborhood has to be tested via the previous boundary detection method for assuring that points may be inside or out of $\mathcal{J}_{\mathbb{H}}$. These points are required to generate a neighboring 'surface' and compute the related normal vector: precision is crucial for the light ray intensity and assesses the rendering quality of the illumination model. The code to compute the normal vector is given below. Let each point be defined by a triplet (`x, y, z`) and that `n, s, w, e` are neighboring points, then:

```
n_v.x = ( e.z - w.z ) * ( dy + dy );
n_v.y = -( s.z - n.z ) * ( dx + dx );
n_v.z = ( dx + dx ) * ( dy + dy ) ;
```

These are the normal vector `n_v` coordinates, later applied to find out the light ray intensity `I` according to the illumination model. Values of `n_v` are normalized[9].

3.7 Comparing the illumination models

The different light management is the most evident issue between the following two figures: ambient parameters being equal, the figure (3.1), rendered by the

[8] Here required to point out to this complicate boundary.

[9] Ranging the intensity `I` between 0.0 and 1.0, they can be easily re-scaled into other color systems. For example, let an indexed palette of 256 (= 2^8) entries; the value of `I` multiplies by the possible palette colors (here, 256) to find the color index: `color_index = (int)( I * 255.0 )` (indexes are zero based.).

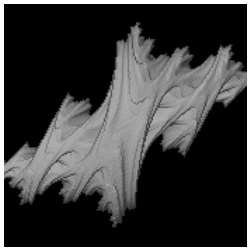


Figure 3.1: Lambertian.

Figure 3.2: Phong.

Lambertian approach, looks flatter than (3.2), processed by Phong shading. Hence one prefers the latter for detailed renderings. Refer to [4] for technical details and code implementation.

4 A new approach: the cut-off rate

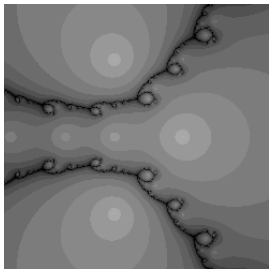
4.1 Limitations of the standard method

Figure 4.1: Plotting the Julia set for iterates of equation (4.3) by the standard method. In figure (4.7), the same was plot by the cut-off rate method.

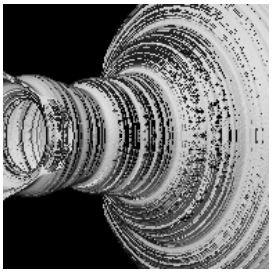
In $\mathbb{C}$, three are the different topologies of Julia sets: connected, totally disconnected and two dimensional (here $J \equiv \widehat{\mathbb{C}}$). The first two cases come up for quaternions too and one wonders if J might be analogously an hypervolume in 4 dimensions filling $\mathbb{H}$ completely[10]. With regard to quaternions, one notices that the standard method is not easily extendable: it just covers the case of $\mathcal{J}_{\mathbb{H}}$ homeomorphic to a closed surface (possibly with curves of multiple points), i.e. a bounded object and surrounded by the basin to ∞: see cases listed at page 8. This always applies to non-linear polynomials because of the attracting point at ∞. By definition, the filled-in $\mathcal{J}_{\mathbb{H}}$ includes all h for which $|f^n(h)|$ is bounded as $n \to \pm\infty$ ([13], p. 71). The inverse iterates, for $n \to -\infty$, suggest the extension required because they[11] might also run through unbounded basins of convergence to finite attracting points. Inside such basins, inverse iterates escape to J which extends to ∞: thus the finite test (hyper-) ball of finite radius becomes obsolete to perform a full boundary scan. The standard method

[10] And, in general, being n-dimensional with regard to the n vectors of the numerical field involved for the generation of J. As far as the author knows, there is no such work devoted to fix this question in details.

[11] As for the iterates of (4.3).



(A)



(B)

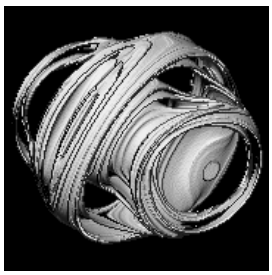
Figure 4.2: (A) shows the iteration of the rational map $\frac{2h^3+1}{3h^2+1}$ in $\mathbb{C}$; (B) shows the same method in $\mathbb{H}$. As predicted from the theory, the second figure can be intended as the 2π-rotation, around the X-axis, of the first one. One also notices that, according to the goal of the extended approach, the interior of the basins was not drawn in (B) in order to evince the Julia set exclusively.

strictly depends on the functionality of such ball (a disc in $\mathbb{C}$) testing if orbits are bounded or they escape to ∞: one implicitly assumes here that there is only one finite attracting point and that its location, together with the related basin, is *a priori* known to lie inside that test ball. In addition, the standard method applies to just one attracting point or to one basin: another limitation concerns the heuristics because, for non filled-in $\mathcal{J}_{\mathbb{H}}$, the previous considerations on the metrics, related to the test ball radius, no longer apply and they are differently accounted to check each iterate. As the problem generalizes for n attracting points δ, one immediately realizes that:

a. checking n trap-balls gets more and more computationally expensive;

b. the locations of all δ cannot be calculated via direct formulas for equations of degree $d > 4$ and thus one cannot count on an *a priori* knowledge: then $(a.)$ would be unapplicable however.

Slice for $h^2 - (0.66 + 0.55i)$.



Julia set for $h^2 - (0.66j + 0.55k)$.

Overcoming these two points will drive to the extension required: once the test ball is dropped off, the method improves and extends to $\mathcal{J}_{\mathbb{H}}$ with zero interior or where the neighboring basins also extend to ∞.

4.2 The extension

It mainly relies on a re-elaboration, in metric terms, of the basics of the theory on normal families[12]. Let p be a point very close to a given Julia set J and let the *point-set distance* be

$$m = \rho(p, J) = \inf\{\rho(p, a) : a \in J\}.$$

Higher order iterates $p_k = f_k(p)$ converge closer and closer to the attracting points δ, so one finds that

$$\lim_{k\to\infty} \rho(p_k, J) \to A = \rho(\delta, J) \tag{4.1}$$

holds *non-uniformly*. The reverse statement can be easily deduced too, assuming non-uniformity again; so, given a value $q > 0$ of the iteration index k for forward[13] images ($k = 0, \ldots, q, \ldots, +\infty$), in general q splits this whole sequence into two subsets so that, metrically speaking, if $k < q$ and for very close p to $\mathcal{J}_{\mathbb{H}}$, the images p_k are closer to $\mathcal{J}_{\mathbb{H}}$ than the forward images $p_k, k \geq q$. This usually rules for orbits entrapped inside filled-in Julia sets: in fact, for bounded basins, $|p_k|$ remain bounded too inside a finite radius disc trapping the orbits and helping to detect $\mathcal{J}$. One solution is to apply a uniform convergence property in the neighborhood of each δ, allowing to get rid *of the location of any* δ (*b.* in §4.1). Orbits will enter a sufficiently close neighborhood $A(\delta, \epsilon)$ as given numerical properties are met, namely when

$$|h_n - h_{n+1}| < \epsilon, \tag{4.2}$$

where $h \in \mathbb{H}, n \in \mathbb{Z}^+$ and a fixed $\epsilon > 0, \lim \epsilon = 0$. Now one needs to 'filter' those iterates being closer and closer to $\mathcal{J}_{\mathbb{H}}$: in this sense, the screen points, associated to quaternion values taking higher and higher iteration indexes to converge and enter $A(\delta, \epsilon)$, shall be plot exclusively. This extension is then defined '*cut-off rate*'.

4.3 Application to iterates

A basins configuration, often arising from rational maps resulting from Newton-Raphson's method applied to an entire $f(h)$, will be examined. For example

[12] Authored by Paul Montel (1876-1975) and playing a fundamental role in the ground-breaking researches by Fatou and Julia in holomorphic dynamics in one complex variable over $\widehat{\mathbb{C}}$, from late 1910s to 1920s.

[13] On the contrary, negative values refer to backward iterates of a map f. They were not deliberately take into account here for sake of ambiguity, because they can be also regarded as forward iterates of inverse map f^{-1}: since the new algorithm does not work on their direct formulas, they have been left out of the discussion. The algorithm itself is based upon the playing with forward iterates exclusively.

let $f(h) : h^3 - 1$, the transformed map is

$$T_f = h - \frac{f(h)}{f'(h)} = \frac{2h^3 + 1}{3h^2}. \tag{4.3}$$

Let f be an holomorphic function, then

$$f^0(z) = z, f^1 = f(z), \dots f^n(z) = f^{n-1}(f(z))$$

are said to be the iterates of rank $0, 1, \dots, n$. The same concept can be applied to the successive iterates T_f^n. In iteration theory, one knows that, for any

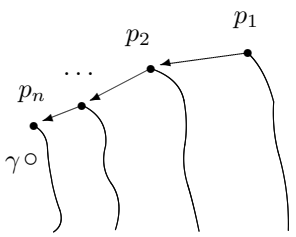


Figure 4.3: Zoom on the neighboring curves $\mathcal{D}_n$, all converging to the attracting fixed point γ.

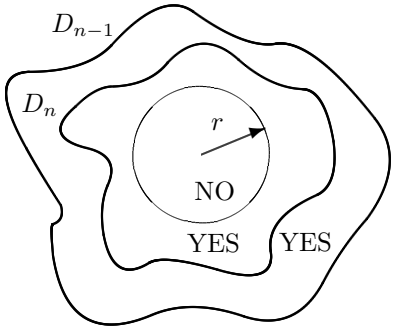


Figure 4.4: Iterates whose distance is smaller than a given value are not considered for the final plot.

attracting fixed point[14] γ_k, the total basin $\mathcal{B}_{\gamma_k}$ of attraction is the union set of all the points whose images, under iterations, converge to γ_k:

$$\lim_{n\to\infty} f^n(z) = \gamma, \qquad \forall z \in \mathcal{B}_{\gamma_k}.$$

Let $\mathcal{D}_1, \mathcal{D}_2, \dots \mathcal{D}_n$, be Jordan curves bounding simply connected domains $D_1, D_2, \dots, D_n$ around γ_k respectively, so that $D_n \supset D_{n+1}$; one would see that, in metric terms, the point-set distance d_n, between the forward iterates $p_1 \in D_1, p_2 \in D_2, \dots, p_n \in D_n$, shrinks to 0 (see figure (4.3)) and one finds again the inequality (4.2): $d_n = |p_n - p_{n-1}| < \epsilon, \epsilon > 0, \lim \epsilon = 0$. So, generally speaking, one can state that, given a seed point $w \in \mathcal{B}_{\gamma_k}$ and close to $\mathcal{J}_{\mathbb{H}}$, the distance $d_n \to 0$ as $n \to \infty$. Conversely, if d^n is constant, then $n \to \infty$ for $|f_n(w) - f_{n+1}(w)| < d^n$ to hold, as w gets closer and closer to $\mathcal{J}$. The goal is to '*isolate*' the points closer to $\mathcal{J}_{\mathbb{H}}$ and plot them exclusively, but with regard to other basins distributions where the test ball, strictly depending on the location, cannot work for granting good results any longer. Figure (4.5) comes from the iterates of (4.3): the basins location, as well as the attracting points,

[14]This extends to such cycles of higher order too: not considered here for sake of simplicity.

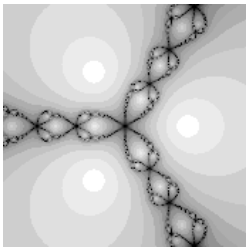


Figure 4.5: Julia set for (4.3) in $\mathbb{C}$.

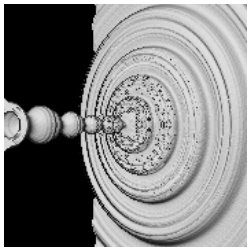


Figure 4.6: The front side in $\mathbb{H}$. The cut on the background is due to the coordinates of the investigation cube.

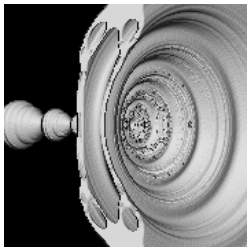


Figure 4.7: The rear side. The grey region at the front accidentally appeared due to the cut for the coordinates of the investigation cube, but helps to find similarities with the analogous experiment in $\mathbb{C}$ and displayed in figure (4.5).

might be out of the ball. According to the theory, $\mathcal{J}_{\mathbb{H}}$ show up as 2π-rotation of the analogues sets in $\mathbb{C}$ and around the X-axis. This approach works like a wall and it is ruled by the condition:

$$|P_n - P_{n-1}| > r, \tag{4.4}$$

where r is the ball radius as in figure (4.4). In C++ coding terms, if `qu` and `next_qu` are two successive iterates with given indexes n and $n+1$ respectively, then the bailout condition of the trapping disc in the standard method:

```
next_qu.abs() <= bailout
```

is finally replaced by

```
next_qu.distancefrom( qu ) >= bailout
```

where the `bailout` is set > 1 in former inequality but ranges from 0 to 1 in the second one. The operator '$\geq$' achieves a reverse task by arresting the orbits out of the ball itself, preventing points inside the ball from further computations, namely the forward iterated domains D_n whose distances is $d_n > r$.

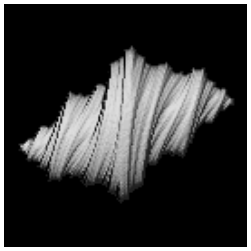


Figure 4.8: Standard method.

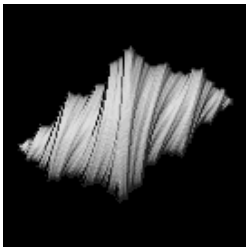


Figure 4.9: Cut-off rate method.

4.4 Considerations

The cut-off rate shall not be confused with the test ball radius. The metrics here involved do not regard the areas of forward domains, but their distances. So only closer points to $\mathcal{J}_{\mathbb{H}}$ are plot, like in figures (4.6) and (4.7). Best graphics are retrieved as the bailout radius r and the maximal iterative index are more finely tuned. This new method is also able to yield good plots for filled-in $\mathcal{J}_{\mathbb{H}}$ (see comparison figures (4.8), (4.9)). Cut-off rate runs a bit slower than the standard one but can be applied for different configurations for $\mathcal{J}_{\mathbb{H}}$.

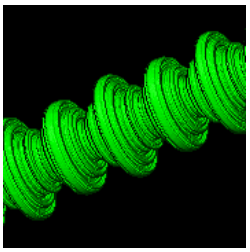


Figure 4.10: Julia set for $h \mapsto \sin(h)$.

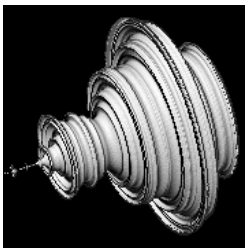


Figure 4.11: Mandelbrot set for $h^2 + c$.

5 The internals of QHD

$\mathbb{R}^3$-submersions are solid models and they just wrap the visible slice of $\mathcal{J}_{\mathbb{H}}$ in order to reduce computation times. In technical terms, the scan splits the given cubic region into a 3-D grid of l^3 sub-cubes and runs through rows and columns, but does not scan the whole cube: it just stops, per each row, when the boundary is met and then restarts from the next row.

5.1 The visual interface

Qhd is an application opening with a user-friendly, small interface to choose the set to be drawn ('Julia' or 'Mandelbrot'), the drawing algorithm ('Standard'

or 'Extended'), a default set of formulas and the chosen illumination method ('*Simple Lambertian*', '*Lambertian*' and '*Phong*'). When properties are displayed, the same window expands and shows additional parameters: the details level (sensibly incrementing the computation times), number of iterates, the degree of predefined maps, the region coordinates and the value of parameter c which works differently as formulas are input to plot Julia or Mandelbrot sets. Input features also list the load/save of metrics related to the displayed object, the bailout value playing as the radius of the disc entrapping the orbits in the standard method or as the value ϵ in the inequality (4.2) of the extended approach. The main windows offers a small preview for quick plots, but one can display a larger magnification window. Additional features include a player to stop the drawing process manually, a window to rotate the given object along the three X-Y-Z axes together with a panel to manage all parameters related to the illumination models (location of light points, intensity, ambient diffusion). The output allows to save the current picture into a graphics file or to be copied into the clipboard and exchange it with other applications.

6 Experiments

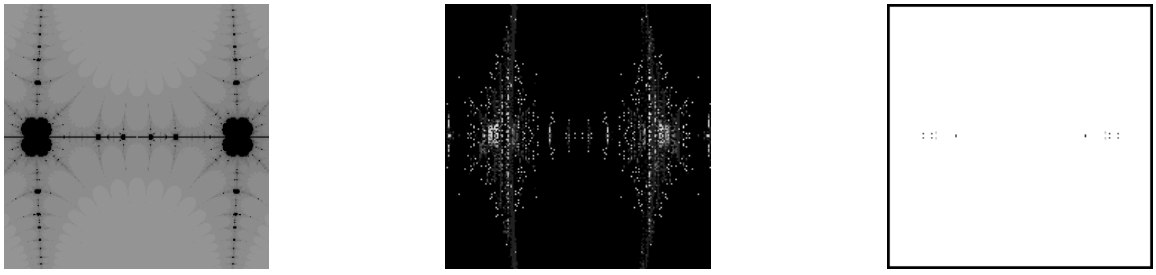
In this section, we try to apply the cut-off rate algorithm and display what happens in $\mathbb{H}$ for other basins distributions which are typical in complex holomorphic dynamics. The same examples will be processed through the standard method, in order to compare the related results.

6.1 Wandering domains

One special configuration occurring for transcendental maps is the wandering domain. In 1983, Sullivan proved that it cannot occur for rational maps.

Definition 1 *Given the Fatou set $\mathcal{F}$, 'Wandering domains' are components $U \subset \mathcal{F}$ such that $f_m(U) \cap f_n(U) = \emptyset$ for any integer $m \neq n$, with $m, n > 0$.*

As empirically shown in (6.1/B), such domains appear for quaternion iterates too. One sees the large filaments in (B) are just due to a little iteration index: for higher order iterates, they will get thinner and thinner but it was deliberately kept small here for enhancing those details which would be hardly visible instead. The largest two nuclei, being clear in the 2-D picture in (A), are less visible in (B) because wrapped by a cloud of 3-D dust arising from the analogous wandering distribution in the submersion of $\mathbb{H}$ into $\mathbb{R}^3$.



(A) (B) (C)

Figure 6.1: Wandering domains in $\mathbb{C}$ and in $\mathbb{H}$ for the function $h + \sin(2\pi h)$. The picture (C) displays (B) as computed by the standard method: results are insufficient here. Colors in (C) have been inverted to set the few points more visible.

6.2 Baker domains and Siegel disks ?

The dynamics of transcendental maps in $\mathbb{C}$ include a basins distribution which cannot occur for rational ones: the 'Baker domain', in honor of his first discoverer Noel Baker. Given the Fatou set $\mathcal{F}$, it is a bounded invariant component $U = f(U) \subset \mathcal{F}$, where there is an essential singularity $a \in \partial U$ such that all iterated orbits inside U converge to a. Displays of Baker domains were not provided here because examples, where the related essential singularity is close to infinity, are the only ones known to the author; since region coordinates are finite, QHD cannot reach any sufficiently close look-up[15]. Siegel discs, whose existence was already conjectured in early 1900s but proven by Carl L. Siegel only in 1942, are components of $\mathcal{F}$ and conformally isomorphic to a disc; the dynamics therein are rotational and the discs come up either for polynomials, rational and transcendental maps. They arise when the first order derivative of the non-linear iterated map is, at the indifferent point, in the form $|e^{2\pi i\theta}| = 1$, where $\theta \in \mathbb{R}\backslash\mathbb{C}$ and enjoys a diophantine condition. Siegel discs in $\mathbb{H}$ are not displayed here due to this early version of the program: the generation of $J_{\mathbb{H}}$ depends on a wrapping around such object, that is, it is not processed and displayed in its full solidity but only from the exterior up to J. The related visualization requires here the dissection of the basin.

6.3 Deforming the Mandelbrot set

The Mandelbrot set $\mathcal{M}_{\mathbb{H}}$ is generated from a parameterized function in the form

$$h_{n+1} = f(h_n, q), \qquad h, q \in \mathbb{H}, \tag{6.1}$$

[15]For example, this comes easier on the Riemann sphere, which is the compactification of $\mathbb{C}$; but there is no analogue trick in $\mathbb{H}$!

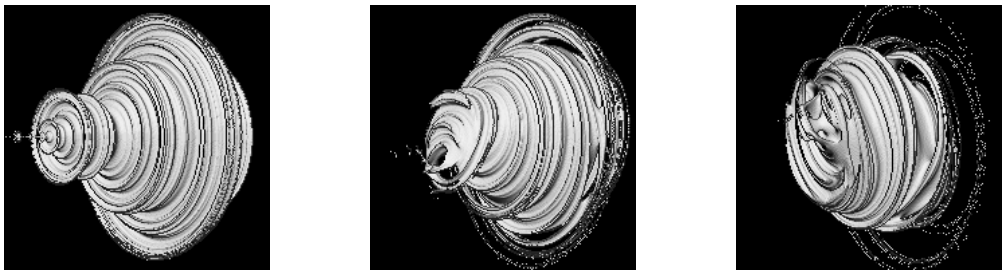
where h_0 is set at one element of the critical points set

$$\mathcal{C} = \{c : f'(c, q) = 0\}$$

of this function itself and q ranges over a given region. One assumes that (6.1) is the quadratic iterator

$$h_n^2 + q. \tag{6.2}$$

In general, if the iterates of (6.1) are bounded, they belong to $\mathcal{M}_{\mathbb{H}}$ and, according to the theory, every critical point c belongs to a basin of attraction. Since, specifically, J from (6.2) are *a priori* known to be totally disconnected or filled-in objects, then each bounded orbit implies that J, related to a given q, is connected and filled-in; otherwise, it is totally disconnected.



(A) (B) (C)

Figure 6.1: (A) shows $\mathcal{M}_{\mathbb{H}}$ when h_0 is the critical point of equation (6.2); h_0 was slightly modified in (B) and (C), producing distortions of the original object.

With the goal of playing with parameters and let them get at any level of computation, QHD can parametrize h_0 too and let it range inside a neighborhood of a critical point c. We noticed that $\mathcal{M}_{\mathbb{H}}$ turns into a 'quasi-$\mathcal{M}_{\mathbb{H}}$' [16]. In the spirit of the experiments in figures (6.1), we just empirically noticed that the quasi-$\mathcal{M}$ sets related to close-to-critical points – see figures (B) and (C) – are not only subjected to distortion, but they are disconnected too. So we wondered if such quasi-$\mathcal{M}_{\mathbb{H}}$ sets are always disconnected, that is, they always show discontinuity regions for the parameter q when h_0 is close-to-critical. If so and with no pretension of deepening this topic, we could pose two questions to advance a strategy to attack the so-called MLC (Mandelbrot set Local Connectivity) conjecture: is $\mathcal{M}_{\mathbb{H}}$ locally connected when $h_0 \in \mathcal{C}$ *exclusively* ? Is it always disconnected if $h_0 \in \mathbb{H} \backslash \mathcal{C}$?

[16] Really this naming convention can be longer applied and a new one should be found because, without the exact value of the critical point, this same computer experiment, on which $\mathcal{M}$ relies, makes no sense anymore.

7 Acknowledgements

I am indebted to Keith Briggs, John Holbrook, Mitja Lakner, Alan Norton and Peter Petek for sending me copies of their works, as well as to Doug Sweetser for helping to improve my code routines computing the quaternion transcendental functions – for example, applied to plot figure (4.10).

8 Conclusions

All images of Julia and Mandelbrot sets in this article have been plot by author's freeware 'QHD', downloadable from `http://www.malilla.supereva.it`.

Alessandro Rosa

Freelance programmer

Brindisi, Italy

`zandor_zz@yahoo.it`

References

[1] Peitgen H.-O., Saupe D., *The Science of Fractals Images*, Springer & Verlag, 1988.

[2] Bedding S., Briggs K., *Iterations of quaternion functions*, American Mathematical Monthly, 103, 1996, pp. 654-664.

[3] Cullen C.G., Duke Math J., 32, 139, 1965.

[4] Foley J.D., van Dam A., Feiner S.K., Hughes J.F., *Computer Graphics, principles and practice*, 2^{nd} edition in C, Addison-Wesley 1997.

[5] Fueter R., Commentarii Mathematici Helvetici, 4, 9, 1932; 7, 307, 1934; 8, 371, 1935-6.

[6] Hart J.C., Sandin D.J., Kauffman L.H., *Ray tracing deterministic 3-D fractals*, Computer Graphics 23 (3), (Proc. SIGGRAPH 89,) July 1989, pp. 289-296.

[7] Hart J.C., Sandin D.J., Kauffman L.H., *Interactive visualization of quaternion Julia sets*, Proc. of Visualization '90, IEEE Computer Society Press, Oct. 1990, pp. 209-218.

[8] Holbrook J.A.R., *Quaternionic Astroids and starfields*, Applied Mathematical Notes, 8 (2), 1983, pp. 1-34.

[9] Holbrook J.A.R., *Quaternionic Fatou-Julia sets*, Annals of Science and Math Quebec, (1), 1987, pp. 79-94.

[10] Kozak J., Petek P., *On the iteration of a Quadratic Family in Quaternions*, V: Mpoyntes, Tasos (ur.), Pneymatikos, Spyros (ur.). Taxe kai chaos. Tomos 4, Polyplokotetas kai chaotikes dynamikes me grammikon systematon, (Dynamika systemata). Athena: Ekdoseis G. A. Pneymatikos, 1998, pp. 121-148. [COBISS.SI-ID 9375833].

[11] Lakner M., Petek P., *Complex and Quaternionic Dynamics – One Equator property*, Exp. math., 1997, let. 6, št. 2, pp. 109-115. [COBISS.SI-ID 7522393].

[12] Mandelbrot B.B., *The fractal geometry of Nature*, W.H. Freeman and Company, New York, 1983.

[13] McMullen C., *Complex Dynamics and Renormalization*, Annals of Mathematics Studies, Priceton University Press, 1994.

[14] Norton A.V., *Generation and rendering of geometric fractals in 3-D*, Computer Graphics 16 (3), 1982, pp. 61-67.

[15] Norton A.V., *Julia sets in the quaternions*, Computers and Graphics 13 (2), 1989, pp. 267-278.

[16] Petek P., *Circles and Periodic points in the Quaternic Julia sets*, Open Sys. & Information Dyn., 4, 1997, pp. 487-492.

[17] Rinehart R.F., Duke Math J., 27, 1, 1960.

[18] Sudbery A., Proc. Camb. Phil. Soc., 85, 1979, 199.