The curvature and dimension of a closed surface

S. Halayka – sjhalayka@gmail.com

October 27, 2019

Abstract

In this short memorandum, the curvature and dimension properties of the 2-sphere surface of a 3-dimensional ball and the $2.x$-dimensional surface of a 3-dimensional fractal set are considered. Tessellation is used to approximate each surface, primarily because the $2.x$-dimensional surface of a 3-dimensional fractal set is otherwise non-differentiable (having no well-defined surface normals). It is found that the curvature of a closed surface *must* lead to fractional dimension.

1 Overview

Unlike in traditional geometry where dimension is an integer (e.g. $(3+1)$-dimensional spacetime), fractional (non-integer) dimension occurs in *fractal* geometry. In fractal geometry, there are currently many ways to calculate the dimension of a surface [1, 2]. This memo uses a new method of calculating the fractional dimension of a surface – it is *curvature* that leads to this fractional dimension.

Our main focus will be on the curvature and dimension of tessellated closed surfaces. For example, Marching Cubes [3] can be used to generate triangular tessellations (meshes), where dimension $D \in (2.0, 3.0)$.

Our attention will be drawn to the difference in curvature and dimension between a 2-sphere and the $2.x$-dimensional surface of a 3-dimensional fractal set. We will generate both a 2-sphere and the $2.x$-dimensional surface of a 3-dimensional fractal set by using iterative quaternion equations.

Some notes are given at the end of this memo.

2 The tessellation of a closed surface

Approximating the surface of a 3-dimensional shape as a mesh allows us to calculate the surface's dimension $D \in (2.0, 3.0)$. This includes approximation of both a 2-sphere and the $2.x$-dimensional surface of a 3-dimensional fractal set.

First we calculate, for each triangle, the average dot product of the triangle's face normal $\hat{n}_i$ and its 3 neighbouring triangles' face normals $\hat{o}_1$, $\hat{o}_2$, $\hat{o}_3$:

$$d_i = \frac{\hat{n}_i \cdot \hat{o}_1 + \hat{n}_i \cdot \hat{o}_2 + \hat{n}_i \cdot \hat{o}_3}{3} \in (-1.0, 1.0]. \tag{1}$$

Because we assume that there are 3 neighbours per triangle, the mesh must be *closed* (no cracks or holes, precisely 2 triangles per edge). The reason why the value -1.0 is not achievable is because that would lead to intersecting triangles.

Then we calculate the normalized measure of curvature, where A_i is the triangle area, and A_{largest} is the largest triangle area in the mesh:

$$k_i = \left(\frac{1 - d_i}{2}\right)\left(\frac{A_i}{A_{\text{largest}}}\right) \in [0.0, 1.0). \tag{2}$$

The triangle area is used to normalize the measure because there are sliver triangles produced by Marching Cubes.

Once k_i has been calculated for all triangles, we can then calculate the average normalized measure of curvature K, where t is the number of triangles in the mesh:

$$K = \frac{1}{t}\sum_{i=1}^{t} k_i = \frac{k_1 + k_2 + ... + k_t}{t} \in (0.0, 1.0). \tag{3}$$

The reason why the value 0.0 is not achievable is because we are dealing with a closed surface, and so there's bound to be *some* curvature.

The dimension of the closed surface is:

$$D = 2 + K \in (2.0, 3.0). \tag{4}$$

As far as we know, this method of calculating the dimension of a closed surface is new. The entire C++ code for generating a mesh can be found at [4]. The entire C++ code for calculating a mesh's dimension can be found at [5].

3 Vanishing versus non-vanishing curvature

Where $r \in [2, \infty)$ is the *integer* sampling resolution, $g_{\max} \in (-\infty, \infty)$ is the sampling grid maximum extent, $g_{\min} \in (-\infty, \infty)$ is the sampling grid minimum extent, and $g_{\max} > g_{\min}$, the Marching Cubes step size is:

$$\ell = \frac{g_{\max} - g_{\min}}{r - 1} \in (0.0, \infty). \tag{5}$$

In this memo $g_{\max} = 1.5$, $g_{\min} = -1.5$, and r is variable.

On one hand, a 2-sphere can be generated by the iterative quaternion Julia set equation

$$Z = Z^2 + C, \tag{6}$$

where the translation constant is $C = 0.0, 0.0, 0.0, 0.0$. For a 2-sphere, the *local* curvature all but vanishes as ℓ decreases (as r increases):

$$\lim_{\ell \to 0.0} K(\ell) = 0.0. \tag{7}$$

This results in a dimension of practically (but never quite) 2.0, which is to be expected from a non-fractal surface. See Figures 1 - 3.

On the other hand, the $2.x$-dimensional surface of a 3-dimensional fractal set can be generated by the iterative quaternion equation

$$Z = Z\cos(Z). \tag{8}$$

For the $2.x$-dimensional surface of a 3-dimensional fractal set, the local curvature does not necessarily vanish as ℓ decreases:

$$\lim_{\ell \to 0.0} K(\ell) \neq 0.0. \tag{9}$$

This results in a dimension considerably greater than 2.0 (but not equal to or greater than 3.0), which is to be expected from a fractal surface. See Figures 4 - 7.

For more information on iterative quaternion equations, and how to perform quaternion multiplication, addition, and cos, see [6].

4 Notes

Hopefully, the following notes and figures are helpful if there is difficulty envisioning the curvature and dimension in the case of Marching Cubes.

4.1 Box-counting dimension for the $2.x$-dimensional surface of a 3-dimensional fractal set

It is simple to obtain the number of boxes that are required to cover the $2.x$-dimensional surface of a 3-dimensional fractal set, thanks to Marching Cubes. One can simply ask: how many triangles are generated per this or that marched cube? If the answer is more than zero, then that particular marched cube covers the surface – the box count $N(\epsilon)$ is increased by one. The box-counting dimension is calculated as usual, and is given in the source code [4]:

$$\dim(\mathrm{S}) = \lim_{\epsilon \to 0.0} \frac{\log N(\epsilon)}{\log(1/\epsilon)}. \tag{10}$$

4.2 Marching Squares

Marching Squares [7] can be used to generate closed line paths, where dimension $D \in (1.0, 2.0)$. For each line segment, the average dot product of the line segment's surface normal $\hat{n_i}$ and its 2 neighbouring line segments' surface normals $\hat{o_1}$, $\hat{o_2}$ is:

$$d_i = \frac{\hat{n_i} \cdot \hat{o_1} + \hat{n_i} \cdot \hat{o_2}}{2} \in (-1.0, 1.0]. \tag{11}$$

Because we assume that there are 2 neighbours per line segment, the line segment mesh must be *closed* (precisely 2 line segments per end point). The reason why the value -1.0 is not achievable is because that would lead to intersecting line segments.

See Figure 8 for an example input (a 2-dimensional greyscale image, consisting of 8x8 pixels) and output (a $1.x$-dimensional closed set of line segments) of the Marching Squares algorithm, approximating a 1-sphere (a circle), where sampling resolution is $r = 8$.

Illustrated in Figure 9 is a section of a closed line path, with surface normals. The average dot product of neighbouring line segments is $d_i = 0.0$. This leads to a normalized measure of curvature $k_i = (1 - d_i)/2 = 0.5$, which in turn leads to an average normalized measure of curvature $K = 0.5$. The dimension is $D = 1 + K = 1.5$.

See Figure 10 for a section of a closed line path as it goes from dimension 1.0 (at top) to 1.9999 (at bottom). In the end, where the dimension is 1.9999, the result is practically a rectangle. The reason why the dimension cannot be 2.0 is because that would lead to intersecting line segments.

4.3 Marching Hypercubes

There is research on Marching Hypercubes at [8, 9]: where dimension $D \in (3.0, 4.0)$, the output is a closed set of tetrahedra. As local curvature all but vanishes, the tetrahedra become as close to regular as possible. For example, where P_{tc} is the tetrahedron centre, P_{fc} is the first face's centre, and P_{ntc} is the first neighbouring tetrahedron's centre:

$$\hat{n_1} = \text{normalize}(P_{tc} - P_{fc}), \tag{12}$$

$$\hat{o_1} = \text{normalize}(P_{fc} - P_{ntc}), \tag{13}$$

and likewise for the other 3 neighbouring tetrahedra, the value of d_i is:

$$d_i = \frac{\hat{n_1} \cdot \hat{o_1} + \hat{n_2} \cdot \hat{o_2} + \hat{n_3} \cdot \hat{o_3} + \hat{n_4} \cdot \hat{o_4}}{4} \in (-1.0, 1.0]. \tag{14}$$

Because we assume that there are 4 neighbours per tetrahedron, the tetrahedral mesh must be *closed* (precisely 2 tetrahedra per face). The reason why the value -1.0 is not achievable is because that would lead to intersecting tetrahedra.

Imagine wrinkly space, begetting curvature, and thus increased dimension.

References

[1] Mandelbrot, B. (1967). "How Long is the Coast of Britain? Statistical Self-Similarity and Fractional Dimension". Science. 156 (3775): 636–8.

[2] Mandelbrot, B. (1982). "The Fractal Geometry of Nature". ISBN 978-0716711865.

[3] Lorensen, W. E.; Cline, Harvey E. (1987). "Marching cubes: A high resolution 3d surface construction algorithm". ACM Computer Graphics. 21 (4): 163–169

[4] `https://github.com/sjhalayka/marching_cubes`

[5] `https://github.com/sjhalayka/meshdim`

[6] Halayka, S. (2009) "Some visually interesting non-standard quaternion fractal sets". Chaos, Solitons & Fractals. Volume 41, Issue 5.

[7] Maple, C. (2003). "Geometric design and space planning using the marching squares and marching cube algorithms". Proc. 2003 Intl. Conf. Geometric Modeling and Graphics. pp. 90–95

[8] Bhaniramka, P; Wenger R.; Crawfis R. (2000) "Isosurfacing in higher dimensions," Proceedings Visualization 2000. VIS 2000 (Cat. No.00CH37145), Salt Lake City, UT, USA, 2000, pp. 267-273.

[9] Bhaniramka, P; Wenger R.; Crawfis R. (2004) "Isosurface construction in any dimension using convex hulls," in IEEE Transactions on Visualization and Computer Graphics, vol. 10, no. 2, pp. 130-141, March-April 2004.

Figure 1: Low resolution ($r = 10$) surface for the iterative quaternion equation $Z = Z^2 + C$, where $C = 0.0, 0.0, 0.0, 0.0$. The surface's dimension is 2.02.

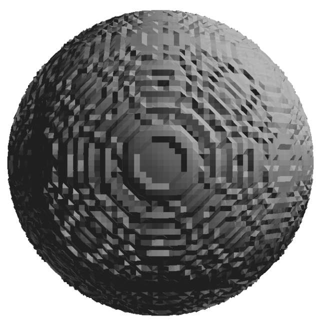

Figure 2: Medium resolution ($r = 100$) surface for the iterative quaternion equation $Z = Z^2 + C$, where $C = 0.0, 0.0, 0.0, 0.0$. The surface's dimension is 2.06.

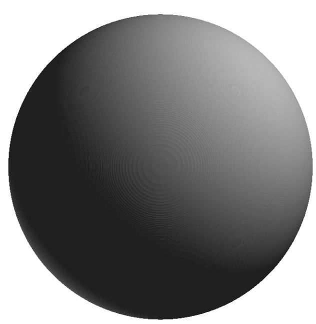

Figure 3: High resolution ($r = 1000$) surface for the iterative quaternion equation $Z = Z^2 + C$, where $C = 0.0, 0.0, 0.0, 0.0$. The surface's dimension is practically 2.0.

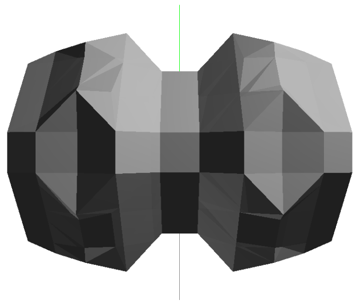

Figure 4: Low resolution ($r = 10$) surface for the iterative quaternion equation $Z = Z\cos(Z)$. The surface's dimension is 2.05.

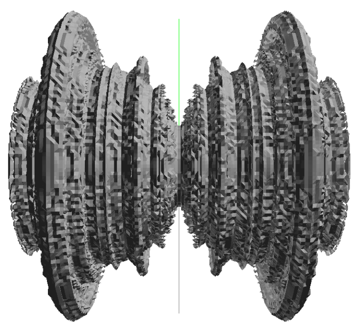

Figure 5: Medium resolution ($r = 100$) surface for the iterative quaternion equation $Z = Z\cos(Z)$. The surface's dimension is 2.11.

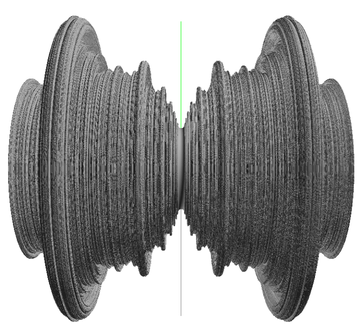

Figure 6: High resolution ($r = 1000$) surface for the iterative quaternion equation $Z = Z\cos(Z)$. The surface's dimension is 2.08.

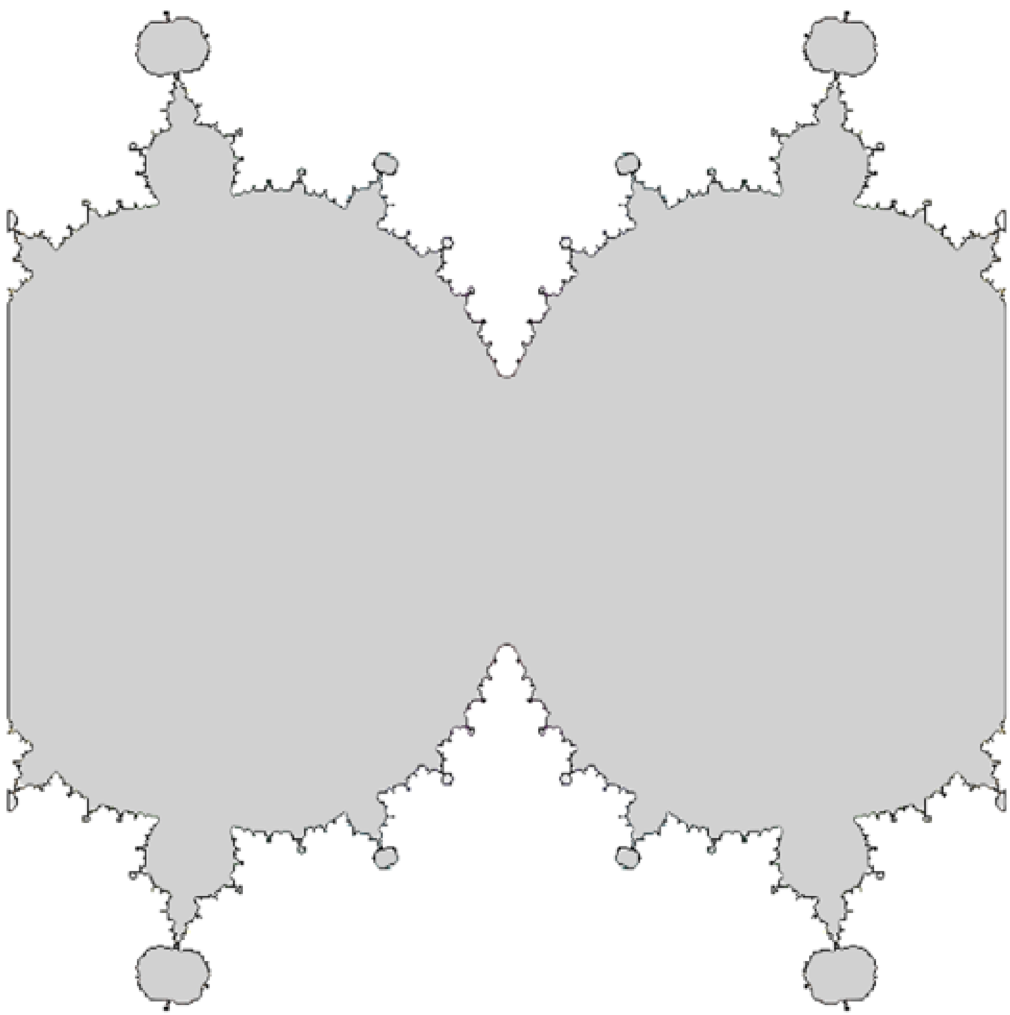

Figure 7: A 2-dimensional slice of the iterative quaternion equation $Z = Z\cos(Z)$, showing the self-similar nature of the set at all scales.

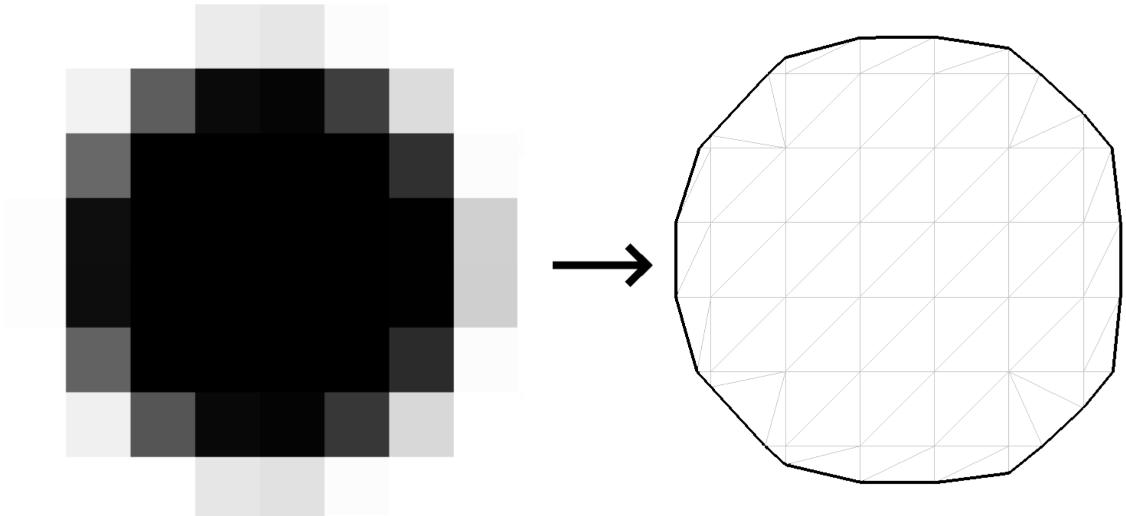

Figure 8: Marching Squares input and output.

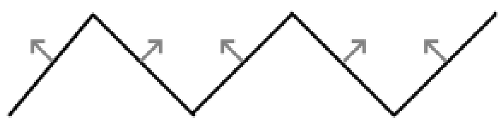

Figure 9: $D = 1.5$.

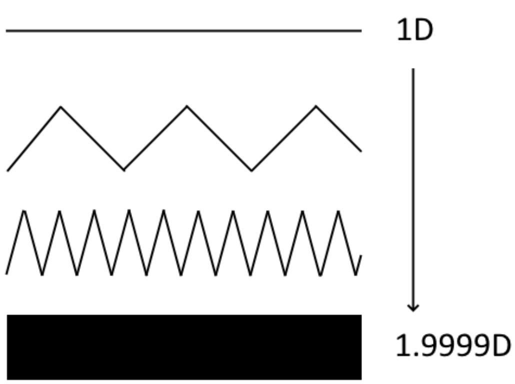

Figure 10: $D = 1$ to $D = 1.9999$.