Visualizing the distributions of the escape paths of quaternion fractals

S. Halayka*

October 15, 2018

Abstract

The length and displacement distributions of the escape paths of the points in some quaternion fractal sets are visualized.

1 Escape path length and displacement histograms

As discussed in [1], a 3D scalar field of quaternion magnitudes $|Z| = \sqrt{Z_x^2 + Z_y^2 + Z_z^2 + Z_w^2}$ results from calculating a quaternion fractal set when using a finite 3D lattice of regularly spaced points as input. Here we will visualize the distributions of the escape paths' lengths and displacements, for those points where $|Z|$ remains below the threshold of 4.0 during 8 iterations. A C++ code is given in the next section, which shows how to perform the iteration process.

See Fig. 1 for a simplified 2D illustration of length and displacement per escape path. The escape paths used to calculate the histograms given in this paper are in 4D.

The histograms in Figs. 2 and 3 together show how the maximum length is generally greater than the maximum displacement. This is also generally the case for the length and displacement per individual escape path, which is generally indicative of curved escape paths – the escape paths generally *meander* because there are *bends*. As for the histograms in Figs. 4 - 7, the maximum length and displacement aren't given, but this does not stop us from using the histograms to visualize the basic properties of the iterative equation in question – the histograms serve as unmistakable (and beautiful) signatures.

In a lot of the cases (but not all cases) a curved escape path forms a loop (see pages 7, 12, and 13 in [2]), which gives rise to the commonly-used name 'orbit' (see [3]). However, most of the time the loop is not quite exact, and so all 'maximum iteration count'+1 = 9 points per escape path end up being distinct. This means that when a curved escape path forms an orbit, the orbit is generally not quite perfect – the curved escape path is likely jittery, or precessing, or spiral-shaped, or all three.

*sjhalayka@gmail.com

2 Core C++ code

The following code performs the iteration process:

```
float quaternion_fractal_set::iterate(
    vector<vector_4> &escape_path_points,
    const quaternion &seed_Z,
    const short unsigned int maximum_iteration_count = 8,
    const float threshold = 4.0f)
{
    // seed_Z constains:
    // 1) a 3D lattice location for seed_Z.xyz, and
    // 2) a constant for seed_Z.w
    Z = seed_Z;

    escape_path_points.clear();

    // Add first point to escape path
    vector_4 p;
    p.x = Z.x;
    p.y = Z.y;
    p.z = Z.z;
    p.w = Z.w;
    escape_path_points.push_back(p);

    // Use squared values to avoid using sqrtf() during the iteration
    float magnitude_squared = Z.self_dot();
    const float threshold_squared = threshold*threshold;

    for(short unsigned int i = 0; i < maximum_iteration_count; i++)
    {
        // Iterative equation
        Z = Z*Z + C;

        // Add additional point(s) to escape path
        p.x = Z.x;
        p.y = Z.y;
        p.z = Z.z;
        p.w = Z.w;
        escape_path_points.push_back(p);

        magnitude_squared = Z.self_dot();

        // Abort early if magnitude tends toward infinity
        if(magnitude_squared >= threshold_squared)
            break;
    }

    return sqrtf(magnitude_squared);
}
```

The following code shows the quaternion implementation:

```
class quaternion
{
public:

    // Constructors omitted for brevity...

    inline float self_dot(void) const
    {
        return x*x + y*y + z*z + w*w;
    }

    quaternion operator*(const quaternion &right) const
    {
        quaternion ret;

        ret.x = x*right.x - y*right.y - z*right.z - w*right.w;
        ret.y = x*right.y + y*right.x + z*right.w - w*right.z;
        ret.z = x*right.z - y*right.w + z*right.x + w*right.y;
        ret.w = x*right.w + y*right.z - z*right.y + w*right.x;

        return ret;
    }

    quaternion operator+(const quaternion &right) const
    {
        quaternion ret;

        ret.x = x + right.x;
        ret.y = y + right.y;
        ret.z = z + right.z;
        ret.w = w + right.w;

        return ret;
    }

    float x, y, z, w;
};

// For iterative equations like Z' = sin(Z) + C *sin(Z)
quaternion sin(const quaternion &in)
{
    quaternion ret;

    const float mag_vector = sqrtf(in.y*in.y + in.z*in.z + in.w*in.w);

    ret.x = sin(in.x) * cosh(mag_vector);
    ret.y = cos(in.x) * sinh(mag_vector) * in.y / mag_vector;
    ret.z = cos(in.x) * sinh(mag_vector) * in.z / mag_vector;
    ret.w = cos(in.x) * sinh(mag_vector) * in.w / mag_vector;

    return ret;
}
```

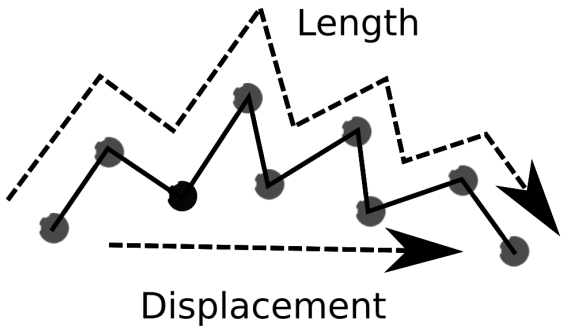

Figure 1: Length and displacement of a meandering escape path that consists of 'maximum iteration count'+1 = 9 points.

References

[1] Halayka S. *Some visually interesting non-standard quaternion fractal sets* Chaos, Solitons & Fractals Vol. 41, Issue 5

[2] `https://iquilezles.org/www/articles/arquimedes/arquimedes.pdf`

[3] `http://math.bu.edu/DYSYS/FRACGEOM/FRACGEOM.html`

[4] `http://www.theworld.com/%7Esweetser/quaternions/intro/tools/tools.html`

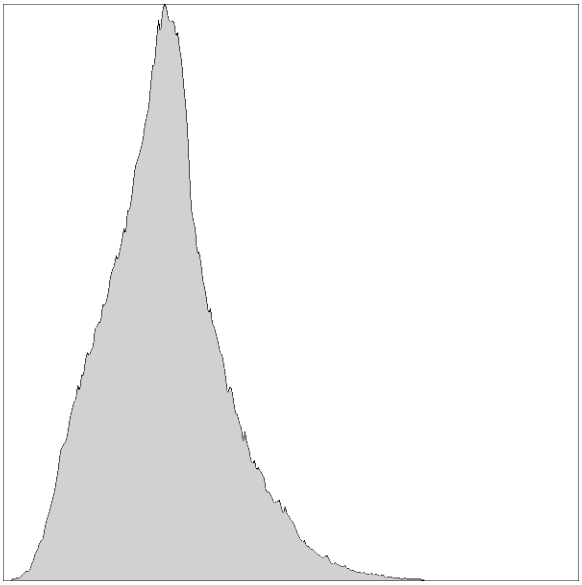

Figure 2: Lengths of $Z' = Z^2 + C$, where $C_{xyzw} = 0.3, 0.5, 0.4, 0.2$. For this histogram the maximum length is 21.2391.

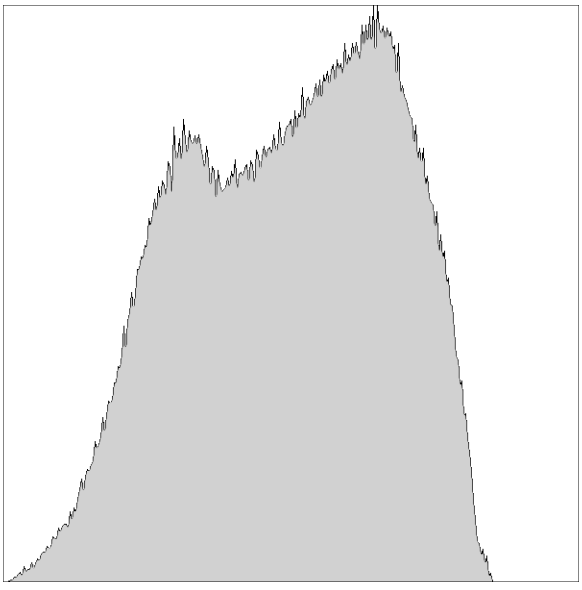

Figure 3: Displacements of $Z' = Z^2 + C$, where $C_{xyzw} = 0.3, 0.5, 0.4, 0.2$. For this histogram the maximum displacement is 2.36506.

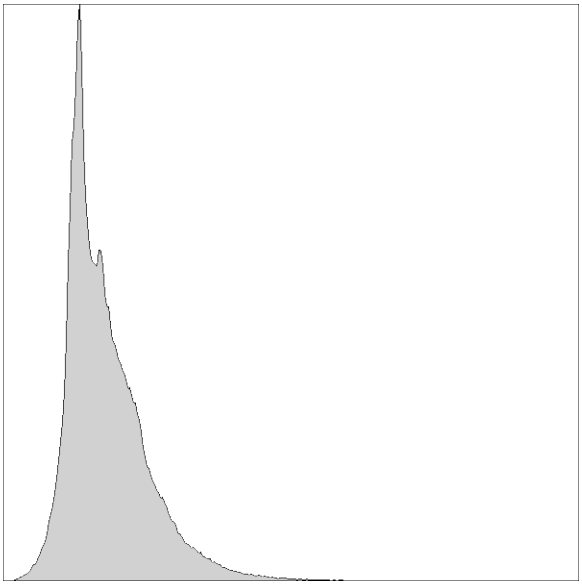

Figure 4: Lengths of $Z' = Z^5 + C$, where $C_{xyzw} = 0.3, 0.5, 0.4, 0.2$.

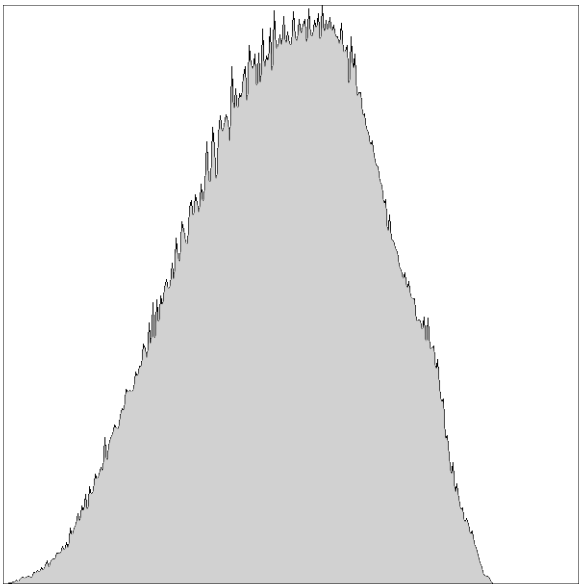

Figure 5: Displacements of $Z' = Z^5 + C$, where $C_{xyzw} = 0.3, 0.5, 0.4, 0.2$.

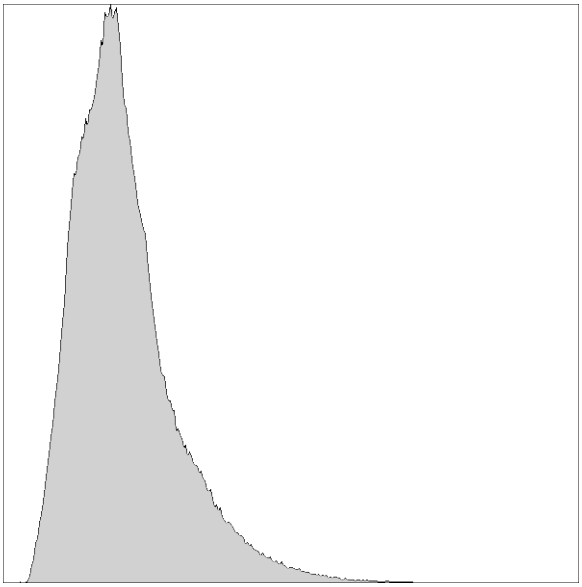

Figure 6: Lengths of $Z' = \sin(Z) + C \cdot \sin(Z)$, where $C_{xyzw} = 0.3, 0.5, 0.4, 0.2$.

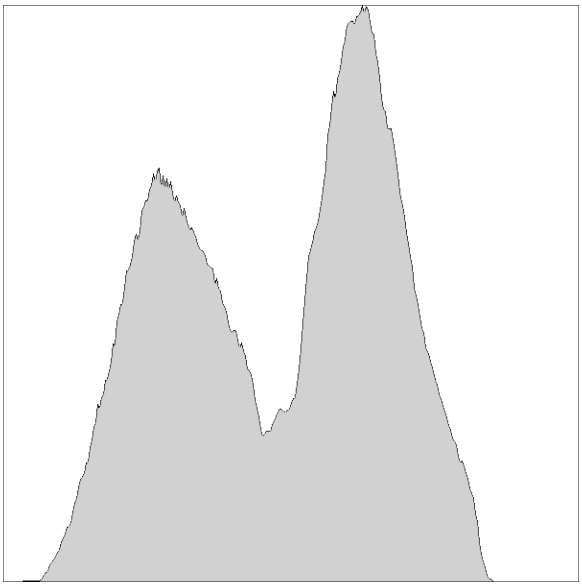

Figure 7: Displacements of $Z' = \sin(Z) + C \cdot \sin(Z)$, where $C_{xyzw} = 0.3, 0.5, 0.4, 0.2$.