Quaternion Language Guide

Version 3.1.1a
February 19, 2023

1. Introduction

What is a Quaternion?

Quaternion is the term used for representing variables with 4 components (a,b,c,d) = a + bi + cj + dk. Quaternions were first described by Hamilton in 1843. Using quaternions make equations very simple and easy to read. It is possible to generate 3D models with quaternions by applying the 3 dimensions of 3D to 3 or more of the 4 components of the quaternion.

What would an equation in Triplex look like?

Writing an equation in triplex would be familiar to anyone that has seen fractal equations in 2D as they look very similar. For example a 2D Mandelbrot looks like:

z = z^2 + cp;

Writing an equation for a 3D fractal using quaternions looks like:

q = q^2 + qp;

The built in variable at the end is different and stands for Complex Point (cp) in 2D and Quaternion Point (qp) using quaternions. Like all Mandelbrot equations, that is the point where the fractal equation is calculated.

Since quaternions have 4 components and 3D space has only 3, the x,y,z of 3D space must be assigned to the components of the quaternion. The default assignment is:

qp = (a , b , c , d) = (x , y , z , 0)

The components may be reassigned in any desired order. See qp in the predefined variables section for more information.

Limitations

Multiplication of quaternions is not commutative.

2. Predefined Variables

Some variables are pre-defined and are used as an interface between the language and the application. The following variables are pre-defined: q, pi, qp, p1-p6, maxit, n, test.

q:

q is the main quaternion variable and is usually iterated. For a Julia type expression, q is initialized to the quaternion value qp. For a Mandelbrot type expression, q is initialized to zero. The magnitude of q is tested to see if it is larger than bailout. If so, the iteration is stopped and q is declared to have escaped. The number of iterations required before q escapes is used to

determine the outer surface of the 3D fractal. The value of q after the last iteration is returned to Fracton but is not used at this time.

Summary:

Julia: q0 = qp; q = q^8 + c; // Common Julia Fractal of q to the 8th power
Mandelbrot: q0 = 0; q = q^8 + qp; // Mandelbrot quaternion fractal of q to the 8th power
Stop iterating q if mag(q) >= bailout

pi:

pi is the quaternion scalar constant (3.14…,0,0,0).

e:

e is the quaternion scalar constant (2.718…,0,0,0).

qp:

qp is the quaternion point in 3D fractal space where the formula will be evaluated. The formula should make no assumption about the limits of qp. The value may be in any direction and may not be visible on any view.

qp = (a , b , c , d) = (x , y , z , 0) Where x,y,z are the 3D fractal coordinates of the point

The components may be reassigned in any desired order by using the component functions and the quaternion (<expression>,<expression>,<expression>,<expression>) function.

Example:
// Make a variable qc that has 3D space assigned as (1 , z , y , x)
// qp is (x , y , z , 0)
qc = (1 , ccomponent(qp) , bcomponent(qp) , acomponent(qp)); // (1 , z , y , x)

p1-p6:

p1-p6 are quaternion values supplied to the formula from the settings view. The values are related to qp and can be used to offset, rotate, and scale qp without any other arithmetic. p1-p6 are also used for animation.

p1-p6 = scale(rotate(qp - position))
p1-p6 = (a , b , c , d) = (x , y , z , 0)

The components may be reassigned in any desired order. See qp.

maxit:

maxit is the maximum number of iterations from the settings view.

n:

n is the iteration number of the loop. n is zero based so the first time through the loop n = 0.

test:

test is the escape test result for the fractal. If test is false, the fractal is declared escaped and the loop stops. If test is true the loop continues as long as the maximum number of iterations has not been exceeded. Use the magsqr function (magnitude squared function) for faster execution than the mag function. The value of test is returned to Fracton after each iteration.

Example: test = magsqr(f) < 4;

3. User Defined Variables

To add a user defined variable, simply use it in an equation. You should set its value before you read its value.

4. Initialization of Variables

Statements before the @beginloop; compiler directive are only executed on the first iteration of the loop.

Example:
q =0; // Initializes f to 0
@beginloop; // Start of the loop
q = q^8 + qp; // Execute until test is false or maxIter is reached
test = magsqr(q) <= 4; // Stop if |q| squared is greater than 4

5. Constants

Constants are quaternion double precision. Exponents are IEEE compliant. To enter a quaternion constant use the quaternion constant function or its shortcut (1,2,3,4) as shown in the example below.

Examples:

a = -3.14e-10; b = 4E10; c = 3; // c is (3,0,0,0)
b = (2,4,6,8);
c =triplex(2,4,6,8);

6. Command Summary

Arithmetic: + - * / ^ %

Assignment: =

Comment: // /* */

Compare: < > <= >= == != && ||

Conditional Expression: ?:

Non-Trigonometric Functions: abs ceil floor mag magsqr pow sqrt

Trigonometric Functions: acos acosh asin asinh atan atan2 atanh cos cosh exp log log10 sin sinh tan tanh. NOTE THAT TRIGONOMETRIC FUNCTIONS ARE NOT AVAILABLE IN THIS FIRST RELEASE OF THE QUATERNION COMPILER.

Quaternion Functions: quaternion acomponent bcomponent ccomponent dcomponent

7. Arithmetic

In all of the descriptions below a and b are triplex numbers. Each has a x, y, and z component. So, the variable a could be represented as (x,y,z).

+ Addition

Adds two quaternion numbers α and β and returns a quaternion number.

$$\gamma = \alpha + \beta$$
$$\gamma.a = \alpha.a + \beta.a$$
$$\gamma.b = \alpha.b + \beta.b$$
$$\gamma.c = \alpha.c + \beta.c$$
$$\gamma.d = \alpha.d + \beta.d$$

Formula example:

f = g+ h;

- Subtraction

Subtracts two quaternion numbers α and β and returns a quaternion number.

$$\gamma = \alpha - \beta$$
$$\gamma.a = \alpha.a - \beta.a$$
$$\gamma.b = \alpha.b - \beta.b$$
$$\gamma.c = \alpha.c - \beta.c$$
$$\gamma.d = \alpha.d - \beta.d$$

Formula example: f = g - h;

Unary minus negates a variable or parenthesized expression.

Formula example: f = -g; f = -(g * 2);

* Multiplication

Multiplies two quaternion numbers α and β and returns a quaternion number. Multiplication is not commutative. Multiplication is performed as a Hamilton product.

$$\gamma = \alpha * \beta$$
$$\gamma.a = \alpha.a * \beta.a - \alpha.b * \beta.b - \alpha.c * \beta.c - \alpha.d * \beta.d$$
$$\gamma.b = \alpha.a * \beta.b + \alpha.b * \beta.a + \alpha.c * \beta.d - \alpha.d * \beta.c$$
$$\gamma.c = \alpha.a * \beta.c - \alpha.b * \beta.d + \alpha.c * \beta.a + \alpha.d * \beta.b$$
$$\gamma.d = \alpha.a * \beta.d + \alpha.b * \beta.c - \alpha.c * \beta.b + \alpha.d * \beta.a$$

Formula example: f = g * h;

/ Division

Divides two quaternion numbers α and β and returns a quaternion number. The reciprocal of the denominator is defined as the conjugate divided by the norm squared. The reciprocal of the denominator is then multiplied by the other component as defined in multiplication above.

$$\beta^{-1} = \frac{\beta*}{\| \beta \|^2}$$
$$= \frac{(\beta.a, -\beta.b, -\beta.c, -\beta.d)}{\beta.a^2 + \beta.b^2 + \beta.c^2 + \beta.d^2}$$

$Then$

$$\gamma = \alpha * \beta^{-1}$$

Formula example: f = g / h;

^ Power

Raises the quaternion number α to the power of the a component of β and returns a quaternion number. The operator ^ is right associative so a ^ b ^ c = a ^ (b ^ c).

$$\| \alpha \| = \sqrt{\alpha.a^2 + \alpha.b^2 + \alpha.c^2 + \alpha.d^2}$$
$$\hat{n} = \frac{(\alpha.b, \alpha.c, \alpha.d)}{\sqrt{\alpha.b^2 + \alpha.c^2 + \alpha.d^2}}$$
$$\varphi = arccos(\frac{\alpha.a}{\| \alpha \|})$$
$$\gamma = \| \alpha \|^{\beta.a}(cos(\varphi\beta.a) + \hat{n}(sin(\varphi\beta.a))$$

Example: f = g ^ h;

% Modulus

Performs the scalar modulus operation on the x component of a triplex number. For the triplex modulus see the mag function. Returns 0 for b <= 0.

g % h
if (h > 0) f.a = g.a % h.a
else f.a = 0
f.b = 0
f.c = 0
f.d = 0

Example f = g % h;

8. Assignment

=

The equal copies the value of a triplex expression to a triplex variable. Every expression must start with a variable followed by = .

f = g

Example: f = g;

9. Comment

//

Any time // is encountered the rest of the line is ignored by the compiler.

Examples:

// This line is a comment
f = g + h; // Everything to the right of // is a comment

/*

The start of a block comment.

*/

The end of a block comment.

Example:

/* All of these three lines
g = g * h;
are comments */

10. Compare

<

g < h returns 1 if g.a is less than h.a. Otherwise, it returns a 0.

Example f = g < h;

>

g > h returns 1 if g.a is greater than than h.a. Otherwise, it returns a 0.

Example f = g > h;

<=

g <= h returns 1 if g.a is less than or equal to h.a. Otherwise, it returns a 0.

Example f = g <= h;

>=

g >= h returns 1 if g.a is greater than or equal to h.a. Otherwise, it returns a 0.

Example f = g >= h;

==

g == h returns 1 if g.a is equal to h.a. Otherwise, it returns a 0.

Example f = g == h;

!=

g != h returns 1 if g.a is not equal to h.a. Otherwise, it returns a 0.

Example f = g != h;

&&

g && h returns 1 if both g.a and h.a are greater than 0. Otherwise, it returns a 0.

Example f = g > 1 && g < 0;

||

g || h returns 1 if either g.a or h.a are greater than 0. Otherwise, it returns a 0.

Example f = g > 1 || g < 0;

11. Conditional Expression:

?:

The conditional expression f ? g : h returns g if f.a > 0 otherwise it returns h. This differs from the C language in that all of the expressions are evaluated whether they are used or not.

Example: q = f >= 0 ? (f - 1) : (f + 1);

12. Non-Trigonometric Functions

abs

Returns the absolute value of the individual components of a triplex number. See mag() for the triplex modulus, triplex absolute value, or triplex magnitude. For the modulus of the x component only see % in the arithmetic section.

abs(g)
f.a = abs(g.a)
f.b = abs(g.b)
f.c = abs(g.c)
f.d = abs(g.d)

Example: f = abs(g);

ceil

Returns the ceiling of the individual components of a triple number.

ceil(g)
f.a = ceil(g.a)
f.b = ceil(g.b)
f.c = ceil(g.c)
f.d = ceil(g.d)

Example: f = ceil(g);

floor

Returns the floor of the individual components of a triplex number.

floor(g)
f.a = floor(g.a)

f.b = floor(g.b)
f.c = floor(g.c)
f.d = floor(g.d)

Example: f = floor(g);

mag

Returns the scalar magnitude of a triplex number. This operation is sometimes referred to as the triplex modulus or triplex absolute value. In math formulas it is often written as |a|.

mag(g)
f.a = sqrt(g.a * g.a + g.b * g.b + g.c * g.c + g.d * g.d)
f.b = 0
f.c = 0
f.d = 0

Example: f = mag(g);

magsqr

Returns the scalar magnitude of a triplex number. This operation is sometimes referred to as the triplex modulus squared or triplex absolute value squared. In math formulas it is often written as |a|^2. This function executes slightly faster than the mag function since it has no square root.

magsqr(g)
f.a = g.a * g.a + g.b * g.b + g.c * g.c + g.d * g.d
f.b = 0
f.c = 0
f.d = 0

Example: f = magsqr(g);

pow

Raises the triplex number a to the power of the x component of triplex number b and returns a triplex number. See the ^ operation above for a code description.

pow(g , h)
// See ^

Example: f = pow(g , h);

round

Returns the round of the individual components of a triplex number.

round(g)
f.a = round(g.a)

f.b = round(g.b)
f.c = round(g.c)
f.d = round(g.d)

Example: f = round(g);

trunc

Returns the trunc of the individual components of a triplex number.

trunc(g)
f.a = trunc(g.a)
f.b = trunc(g.b)
f.c = trunc(g.c)
f.d = trunc(g.d)

Example: f = trunc(g);

sqrt

Returns the square root of a triplex number. This is implemented by raising a triplex number to the exponent of 0.5 using the power operation.

sqrt(g)
f = g^0.5 // See ^

Example f = sqrt(g);

13. Trigonometric Functions

The trigonometric functions are: acos acosh asin asinh atan atan2 atanh cos cosh exp log log10 sin sinh tan tanh. IN THE INITIAL RELEASE ONLY COS, EXP, LOG, AND LOG10 ARE FUNCTIONAL.

cos

Returns the cosine of a quaternion number α as a quaternion β.

$$\beta = cos(\alpha)$$
$$\| \alpha \| = \sqrt{\alpha . a^2 + \alpha . b^2 + \alpha . c^2 + \alpha . d^2}$$
$$\hat{v} = (\alpha . b, \alpha . c, \alpha . d)$$
$$\| \hat{v} \| = \sqrt{\alpha . b^2 + \alpha . c^2 + \alpha . d^2}$$
$$\hat{n} = \frac{\hat{v}}{\| \hat{v} \|}$$
$$\beta = cos(\alpha . a)cosh(\| \hat{v} \|) - \hat{n}sin(\alpha . a)sinh(\| \hat{v} \|)$$

Formula example: f = cos(g);

exp

Returns the exponential of a quaternion number α as a quaternion β.

$$\beta = exp(\alpha)$$
$$\| \alpha \| = \sqrt{\alpha . a^2 + \alpha . b^2 + \alpha . c^2 + \alpha . d^2}$$
$$\hat{v} = (\alpha . b, \alpha . c, \alpha . d)$$
$$\| \hat{v} \| = \sqrt{\alpha . b^2 + \alpha . c^2 + \alpha . d^2}$$
$$\hat{n} = \frac{\hat{v}}{\| \hat{v} \|}$$
$$\beta = e^{\alpha . a}(cos(\| \hat{v} \|) + \hat{n}sin(\| \hat{v} \|))$$

Formula example: e = exp(f);

log

Returns the natural logarithm of a quaternion number α as a quaternion β.

$$\beta = ln(\alpha)$$

$$\| \alpha \| = \sqrt{\alpha . a^2 + \alpha . b^2 + \alpha . c^2 + \alpha . d^2}$$

$$\hat{v} = (\alpha . b, \alpha . c, \alpha . d)$$

$$\| \hat{v} \| = \sqrt{\alpha . b^2 + \alpha . c^2 + \alpha . d^2}$$

$$\hat{n} = \frac{\hat{v}}{\| \hat{v} \|}$$

$$\beta = ln(\| \alpha \|) + \hat{n} arccos(\frac{\alpha . a}{\| \alpha \|})$$

Formula example: e = log(f);

14. Quaternion Functions

quaternion

Returns a quaternion number from the a component of four quaternion numbers.

quaternion(e, f, g, h)
q.a = e.a
q.b = f.a
q.c = g.a
q.d = h.a

Formula example
e = (1,2,3,4);
f = (5,6,7,8);
g = (9,10,11,12);
h = (13,14,15,16);
q = quaternion(e, f, g, h); // q is (1,5,9,13)

acomponent

Returns the a component of a quaternion number α as a quaternion number β.

β = acomponent(α)
β.a = α.a
β.b = 0
β.c = 0
β.d = 0

Formula example
g = (1,2,3,4)
f = acomponent(g); // f is (1,0,0,0)

bcomponent

Returns the b component of a quaternion number α as a quaternion number β.

β = bcomponent(α)
β.a = 0
β.b = α.b
β.c = 0
β.d = 0

Formula example
g = (1,2,3,4)
f = bcomponent(g); // f is (2,0,0,0)

ccomponent

Returns the c component of a quaternion number α as a quaternion number β.

β = ccomponent(α)
β.a = 0
β.b = 0
β.c = α.c
β.d = 0

Formula example
g = (1,2,3,4)
f = ccomponent(g); // f is (3,0,0,0)

dcomponent

Returns the d component of a quaternion number α as a quaternion number β.

β = dcomponent(α)
β.a = 0
β.b = 0
β.c = 0
β.d = α.d

Formula example
g = (1,2,3,4)
f = dcomponent(g); // f is (4,0,0,0)

15. Precedence:

( )

Changes the precedence of an expression.

Example: a = 2 * (1 + 3); // a is (8,0,0,0)

16. Control Flow

if, if else, and else are used to allow logical control of program execution.

if

The if operator must be followed by an expression and one or more statements enclosed by brackets. If the expression is not equal to zero the expression is true and the statements are executed. Unlike C, the brackets are required around statements.

if(expression) {one or more statements}

else if

The else if operator is executed only if the previous if or else if operators were false. The else if operator must be followed by an expression and one or more statements enclosed by brackets. If the expression is not equal to zero the expression is true and the statements are executed.

else if(expression) {one or more statements}

else

The else operator is executed only if the previous if or else if operators were false. The else operator must be followed by one or more statements enclosed by brackets.

else {one or more statements}

Example
```
if( a < 5) {
a = 0;
b = b + 1;
}
else if( a > 10){
a = 0;
}
else {
a = 10;
b = 0;
}
```